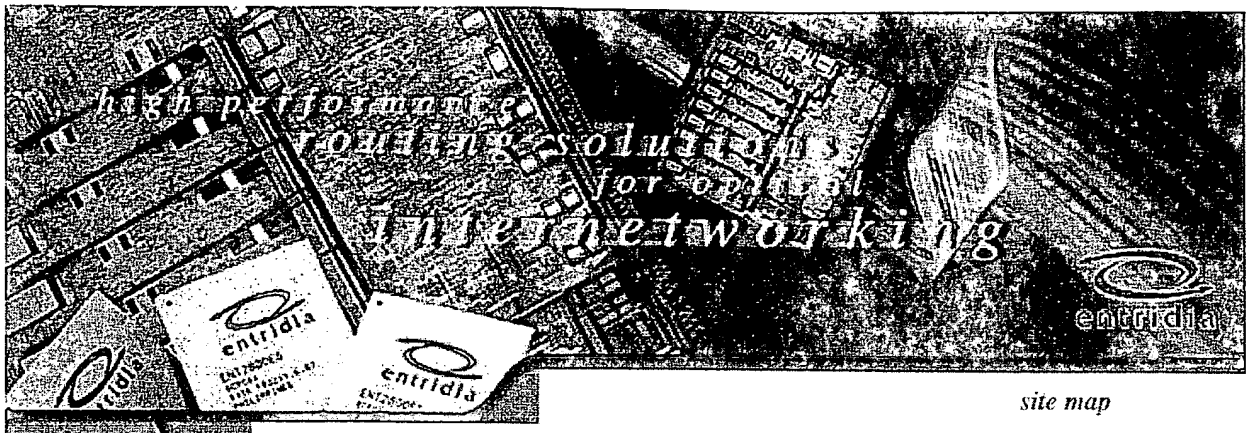


site map

home
about us
technology
products
support
jobs
news & events
partners
contact us

technology

Architecture and Design of Function Specific Wire-Speed Routers for Optical Internetworking

Introduction

Internet traffic has grown dramatically over the last 24 months. Figure 1 chronicles and forecasts the rate of Internet traffic over the next 24 months. A key element that drives the rate of growth of Internet Traffic is the Internet Service Model. While voice and video are seen as emerging real time services, secure commerce transactions and virtual private networks are already being widely deployed and are transforming the underlying mechanics business. As with any disruptive technology, the Internet, brings with it a significant technology challenge - scaling the network infrastructure. The Internet is knit together by Routers. These devices are considered the "neurons" of the network, and seamlessly integrate large sub-networks together using the Internet Protocol. This article focuses on a Network and System level view of high performance routers and attempts to expose the reader to design tradeoffs and approaches to solving the problem of delivering wire-speed Quality of Service that is scalable to WDM rates.

PACKET SIZE

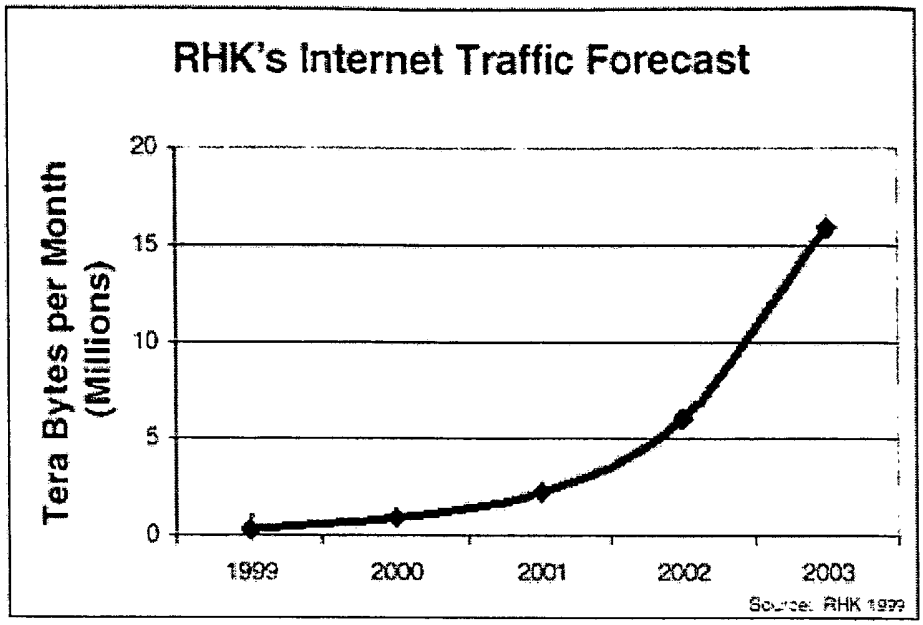


Figure 1. RHK's Internet Traffic Forecast

Driving Forces Behind the new IP infrastructure

Traffic Characteristics

Internet Backbone Traffic: A Quantitative Analysis

A snapshot of Internet backbone traffic profile is shown in Figure 2. From the chart it is clear that the distribution of packet sizes is almost trimodal, with a good 30% of the packets falling into the 40 byte category. These small packets are usually TCP acknowledgment messages, and are appropriately considered to be an important corner case in the design of high performance routers. It is quite conceivable that a traffic flow can consist of back to back minimum size packets sustained over many seconds, and can thus put a considerable amount of "strain" on a router.

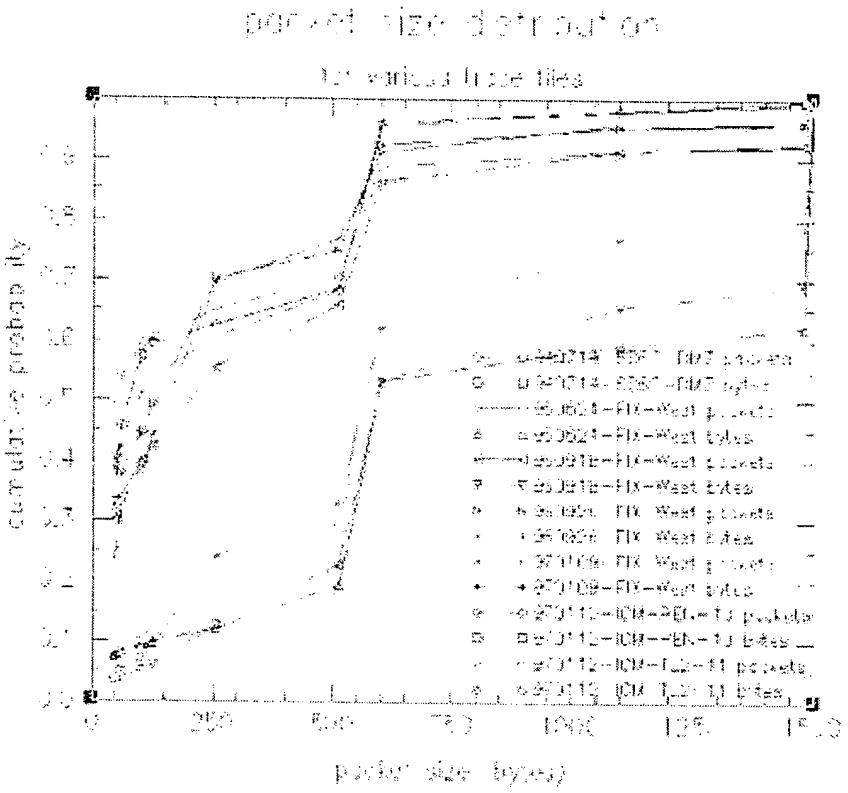


Figure 2. Packet Size Distribution

The Future of Internet Traffic

Future extrapolation of the traffic model can be achieved by combining the forecasted increase in traffic with the trimodal packet size distribution. In the simplest case, we can expect an overall increase in the volume of acknowledgement traffic (min. size packets). In the more realistic case, we must overlay best efforts and mission critical traffic along with streaming services; and consequently need to add a third dimension to the mix of traffic - the type of service.

The Carrier Perspective

New Set of System Metrics

With the move towards a converged IP based infrastructure, Carriers require intelligent devices at network aggregation points. With delay sensitive streaming content being injected into the network, line-rate performance is a non-negotiable basic criterion. Additionally, requirements for ultra-high connection densities, as well as highly efficient (low power), space conscious equipment has led to a new metric to qualify Carrier Gear:
Routed Gigabits per (Power Unit X Rack Space Unit X $)

Fundamental Concepts

Wire-Speed

Wire Speed (or Line-Rate) refers to the maximum rate at which any physical medium can sustain information transfer. The key variables that determine this are the number of bits per second that the physical medium is capable of transporting, as well as the size of the minimum quanta of information (packet or cell). Thus, a link capable of supporting 2.4 Gbps (OC-48c) which carries 40 byte (320-bit) packets with no inter -packet gap or overhead bits corresponds to a packet rate of 1 packet arriving every 129 ns. Wire-speed or line-rate processing requires that operations be performed on a per packet basis at the maximum packet arrival rate (every 129 ns for the case mentioned earlier). Guaranteeing line-rate packet processing and forwarding performance has numerous positive side-effects in the Quality of Service Domain which will be examined later.

The Network Layer

The concept of Routing centers on using Network Layer information to forward packets. The basic Network Layer Functions (OSI Layer 3 and 4) consist of the following:

1. Route Processing:	Where is the packet destined ?
2. Flow Processing:	Stateful information that categorizes a packet or group of packets that belong to an information session.
3. Path Discovery:	Stateful knowledge of the network topology and changes associated with it.

Route Processing

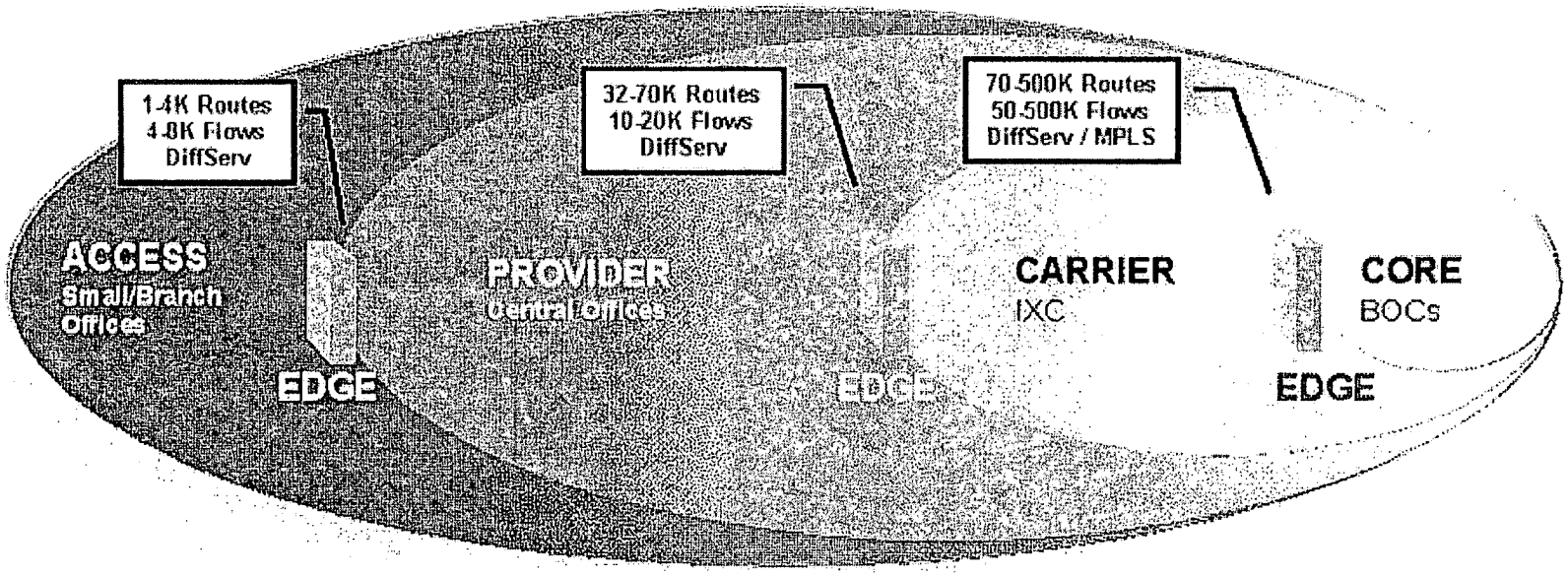
The action of determining the destination of a packet based on data embedded within it is termed Route Processing. IPv4 networks use Classless Inter Domain Routing (CIDR) which was instituted by the IETF in the 1980's to optimize the usage of available address space. The basic principles of CIDR involve the segmentation of the Internet into a hierarchical, logically addressable group of sub-networks. Consequently, each router is required to only keep track of paths that are directly accessible via its network interfaces. CIDR's logical addressing scheme requires a longest network prefix match operation, which is set by a mask on a 32 bit IPv4 address. CIDR route lookups are not direct table matches, and thus become quite complex with large tables.

CIDR Route Lookup challenges

The complexity of a CIDR route lookup dramatically changes with the total number of routes in a route table. The nested nature of the addressing scheme causes a logarithmic change in lookup time with increased table size. Wire speed algorithmic CIDR route lookup is nontrivial, as it involves translating an algorithm into hardware (ASIC); and ensuring that it provides deterministic convergence under worst case traffic conditions. A second challenge is to keep the jitter i.e. variation in algorithm convergence timings, bounded so as to bound latency within the network.

Flow Processing

Packet classification is the key element of flow processing. Packets may be classified based on a parametrized set of metrics that may involve multi-field packet header analysis. The parameters are usually specified by a user in


1-4K Routes
4-8K Flows
DiffServ
32-70K Routes
10-20K Flows
DiffServ
70-500K Routes
50-500K Flows
DiffServ / MPLS
ACCESS
Small/Branch
Offices
EDGE
PROVIDER
Central Offices
EDGE
CARRIER
IXC
CORE
BOCs
EDGE

conjunction with resource information that may be derived from routing protocols.

What constitutes a flow?
Flows in connectionless networks are determined by grouping packets that have common application layer or session layer information. A flow can be based on information transacted between a particular Source and Destination IP address, or a TCP/UDP socket. Flows can also be based on DiffServ code points or Type of Service bits. Fundamentally, classification of like packets based upon information contained within each of them constitutes a Flow.

The role of DiffServ
Differentiated Services results from IETF initiatives to specify a means of providing end to end QoS in a connectionless packet based network. The IPv4 packet header comprises a byte which consists of a 3 bit Type of Service field and a 5 bit field that provides 32 extra code points for marking packets to denote various levels of service. These DiffServ labels may be generated from source nodes in the network, and may be altered by intermediate routers to shape network traffic. DiffServ is meant to provide a granular means of differentiating classes of service at the network edge.

Macro vs. Microflows
As mentioned earlier, a flow can be identified by various parameters including DiffServ labels, and application (TCP/UDP) information. Edge flows with granularity are termed Microflows. An example of a microflow would be the classification of all packets of a certain TCP/UDP socket that originate from a particular IP address; or all RTP traffic destined for a certain IP address. Once a packet has been classified at the network Edge and has been identified with a particular flow, it is forwarded out of the particular routing device onto the next level of aggregation within the network. Fig. 3 shows a multi-edge Internet model that illustrates the various levels of aggregation occurring at different points in the network and the rough route and flow metrics at these points. It is important to recognize the inefficiency of multiple examinations of the same flow of packets at various aggregation points. In fact, as we approach the backbone, data pipes get larger and packet arrival rates increase, making it impractical to perform deep packet examination within the core. Additionally, the core may be operating on a different link-layer protocol - such as ATM. Enter Macroflows. A Macroflow consists of a logical grouping of similar Microflows. For instance, all packets entering a backbone or core device that have similar microflow information (ex: DiffServ labels) may be grouped into a Macroflow; and can be metered, policed and engineered efficiently. The concept of hierarchy in flow management and QoS classification has lead to the use of Multi-Protocol Label Switching (MPLS) as a means to manage and engineer Macroflows.

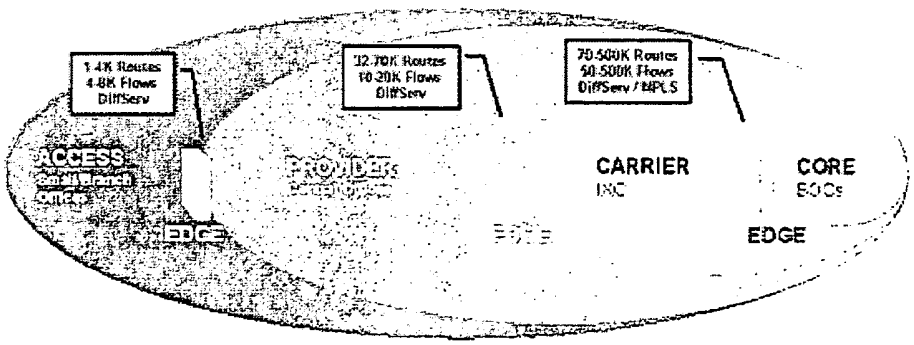


Figure 3. Internetscape - Performance View
(Click on image for a larger version)

The role of MPLS
MPLS (Multi Protocol Label Switching) was initially conceived to be a mechanism that unified the IP and ATM domains at the Internet Core. It has, however, also become a powerful traffic engineering tool. At the simplest level, MPLS allows core traffic to be engineered at either a circuit level via an ATM switch; or at a packet level. The actual physical tag may denote an ATM virtual channel which has prescribed traffic behavior; or may be used as a way to abstract Layer 3 Microflow information and engineer macroflows at the core.

Path Discovery

Routing protocols such as RIP, OSPF or BGP -n are inter-router information exchange mechanisms that build and maintain packet forwarding tables which are used by the packet forwarding blocks to physically route traffic; and by policy and flow software to maintain and update flow tables. These protocols include algorithms that use value metrics based on a variety of parameters. An example would be a network distance-vector metric, i.e. what is the closest network entity that has a path to the final destination. Other metrics that are used to build tables include latency and reliability.

Quality of Service and Traffic Engineering

In an IP network, the Network Layer functions drive the Quality of Service assigned to various types of traffic. QoS is applied via Traffic Engineering which involves three distinct mechanisms:

Admission Control
This mechanism acts on incoming traffic that has been categorized by the Network Layer to ensure that all flows of information meet predetermined profiles (arrival rates) which in turn are determined by Service Level Agreements.

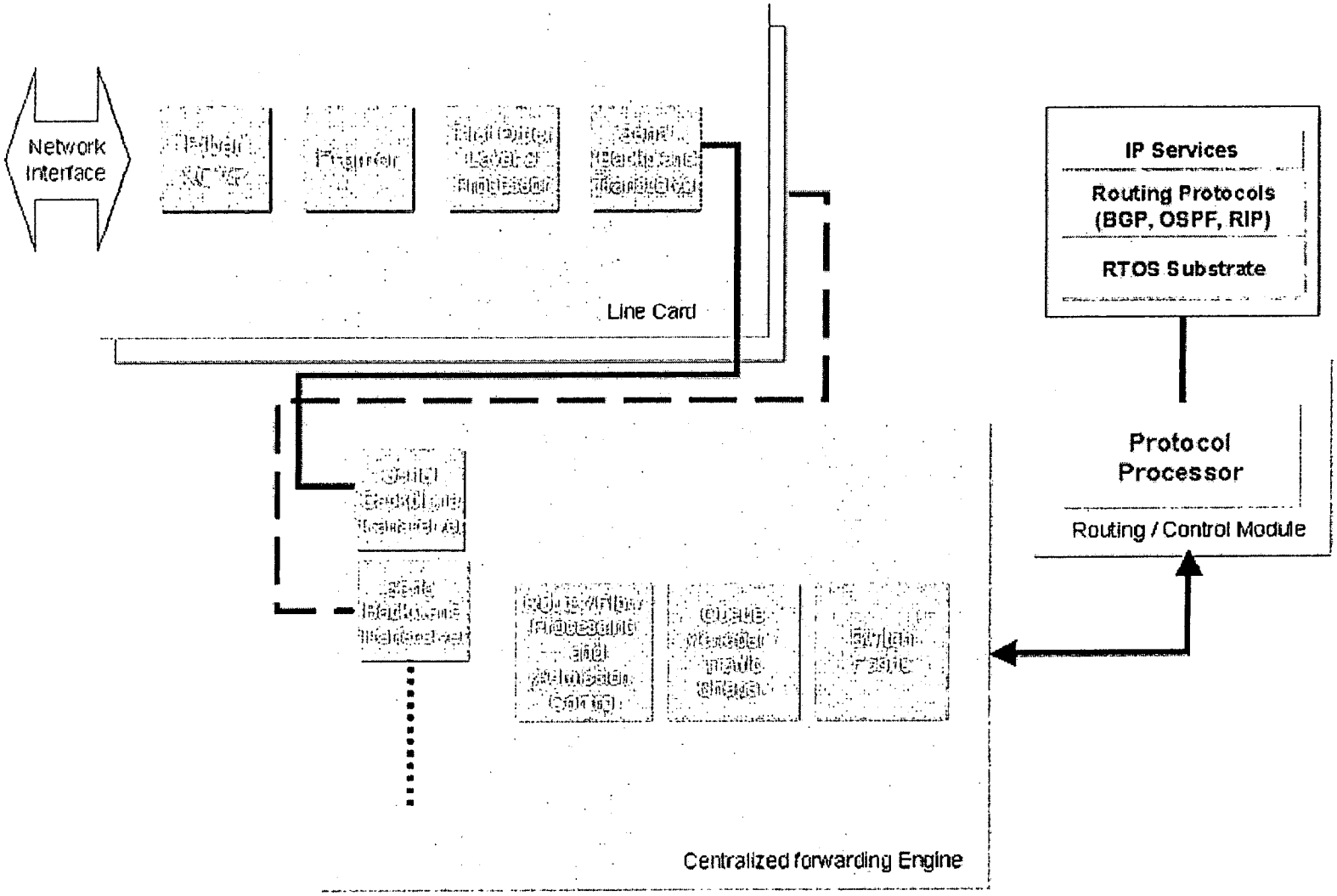
Traffic Shaping and Bandwidth management:
In this case, flows, and other related parameters are used to determine, when, and at what rates various types of packets egress the system. Queuing becomes an essential part of the shaping and bandwidth management.

Congestion Control:
All Network devices are bound to experience congestion. While QoS is generally thought of in terms of prioritizing outgoing traffic, the avoidance of congestion is a key mechanism that is often sidelined or forgotten. Large, time varying traffic patterns coupled with service overlays on the infrastructure could potentially cause network outages. Controlling congestion involves statistical coloring of traffic based on Network and Application layer information. Usually, processes such as RED (Random Early detection); monitor the state of various queues within the system, and start to drop packets based on their fullness. It is important to note that drop processes such as RED can be modulated by weights that are user supplied.

System Architecture

Overview

Fundamentally, a high performance router can be subdivided into two pieces - The routing / path discovery plane; and the packet forwarding plane. These two distinct pieces are subject to various levels of implementation and


Network Interface
Line Card
IP Services
Routing Protocols (BGP, OSPF, RIP)
RTOS Substrate
Protocol Processor
Routing / Control Module
Centralized forwarding Engine

partitioning depending on the router's position within the network. Figure 4 illustrates the typical architecture of a high performance router. It is extremely important to note the path discovery process time constant is on the order of 10 to a 100 milliseconds, while that of the packet forwarding process scales with line-rates (1 / 129 ns at OC-48c). The large time constant difference between these processes presents a logical opportunity for first order partitioning -- separation of the packet classification and forwarding paths from the routing control path. Subsequent architectural decisions involve further partitioning the packet classification / forwarding paths. There are two broad methods that are generally followed: Centralized Packet Forwarding; and Distributed Packet Forwarding. Key factors that drive the choice of approach include scalability, protocol support and power.

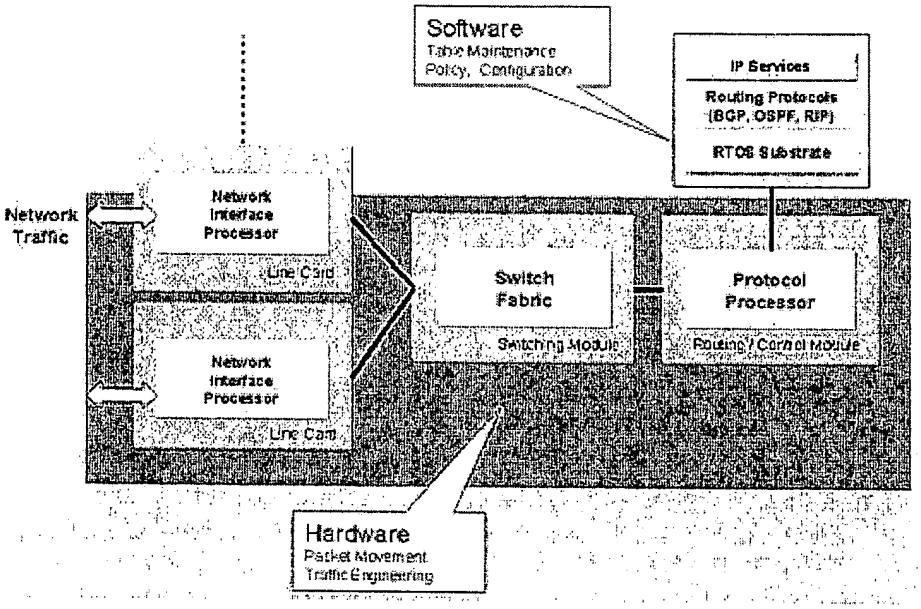
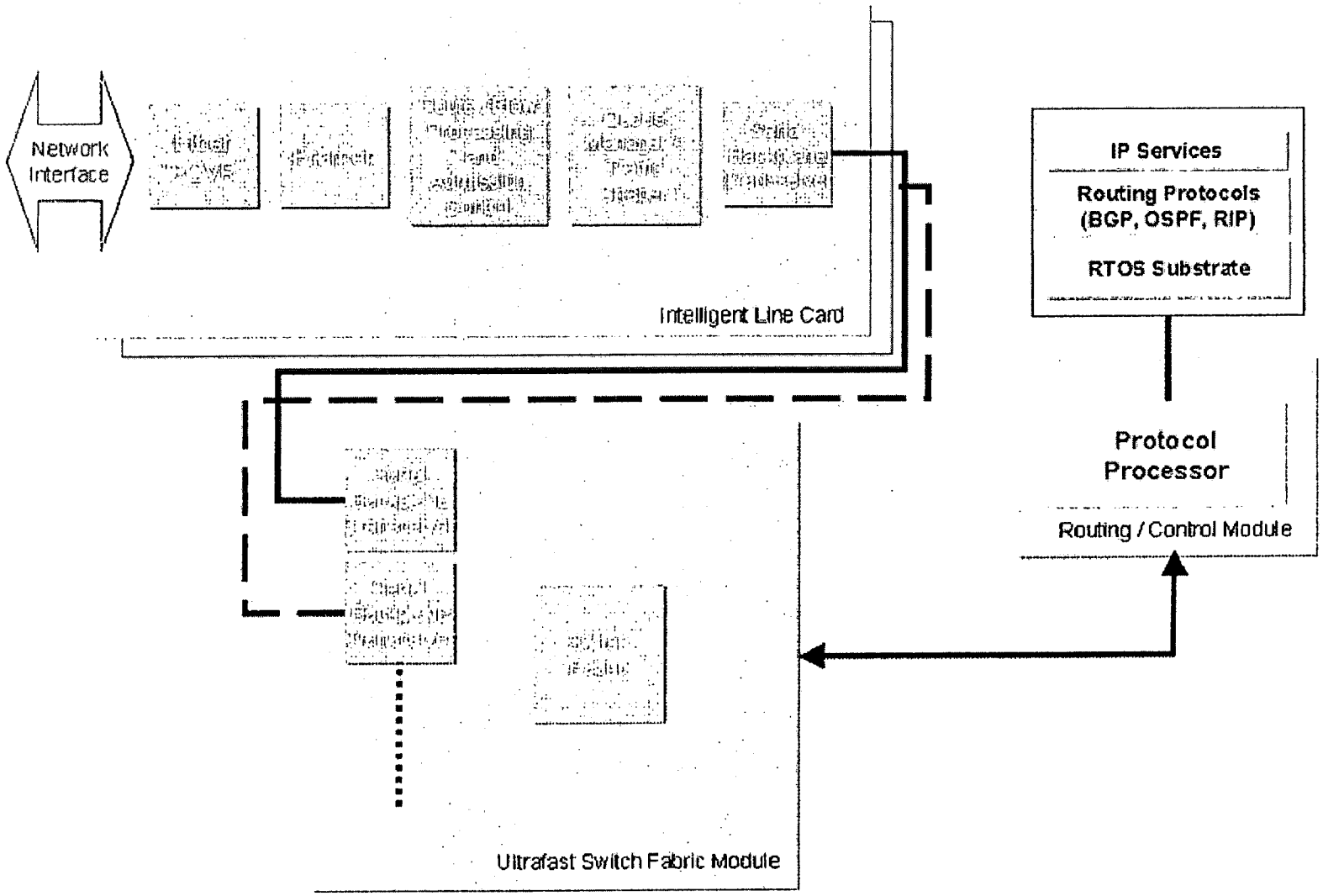


Figure 4. Router Anatomy
(Click on image for a larger version)

Centralized Packet Forwarding

The basic concept here is illustrated in Figure 5. Network Interface Line Cards comprise of PMD (physical media dependent functions), and Link Layer functions (Framers / SAR's); limited packet processing (Layer 3) and in the simplest case directly interface to a Switch Fabric. The Switch Fabric becomes the crucial element in this architecture, as it handles the bulk of the network layer driven Quality of Service / Traffic Engineering Functions as well as basic packet transport across line cards.


Network Interface
Intelligent Line Card
IP Services
Routing Protocols (BGP, OSPF, RIP)
RTOS Substrate
Protocol Processor
Routing / Control Module
Ultrafast Switch Fabric Module

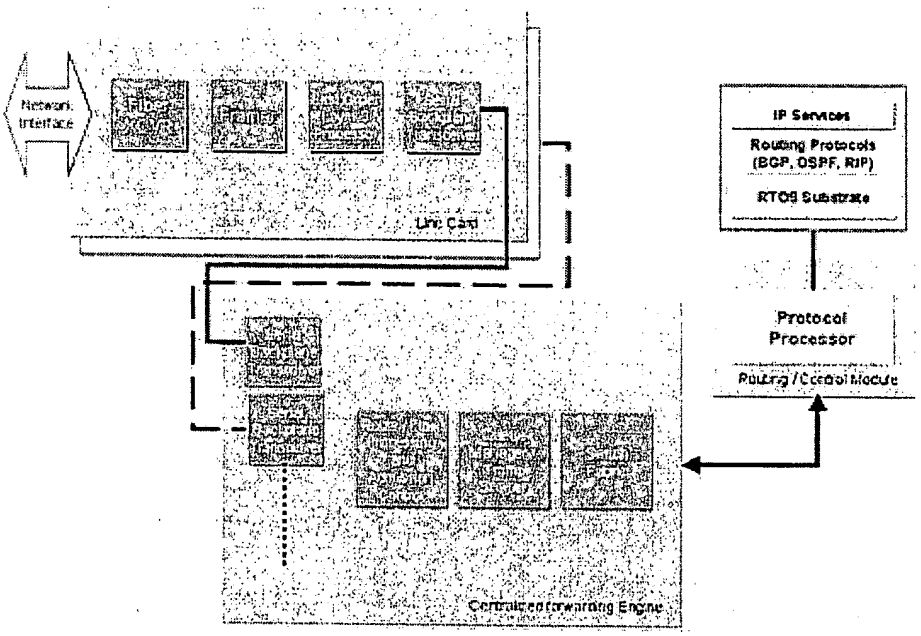


Figure 5. Centralized Packet Forwarding
(Click on image for a larger version)

Distributed Packet Forwarding

The basic concept is illustrated in Figure 6. Network Interface Line Cards essentially assume the role of a full router. They comprise of all the hardware including the PMD and Link Layer, but are also fully equipped with packet processing functionality as well as local Quality of Service and traffic engineering. The Switch Fabric is purely optimized for non-blocking transport of packets across line cards and DOES NOT include any sophisticated Quality of Service / Traffic Engineering functions.

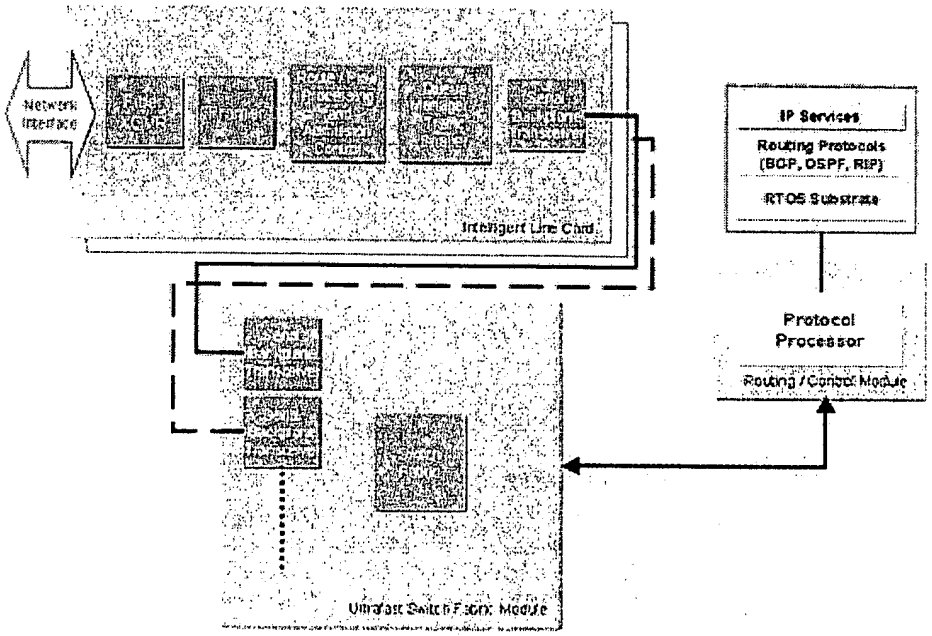


Figure 6. Distributed Packet Forwarding
(Click on image for a larger version)

Summary and Implications

	Centralized Packet Forwarding	Distributed Packet Forwarding
Scalability	Constrained by Scaling the Switch Fabric which comprises the QoS and traffic engineering pieces, which are typically the hardest to scale.	Switch Fabric is optimized for scalable data transport. QoS and traffic engineering is distributed in the line-card - simpler to scale.
Reliability	Single point of failure point rests in the Switch Fabric / Traffic engineering blocks. Redundancy and reliability require replication of this entire function.	No single point of failure. Each card is a mini router. Redundancy can easily achieved by adding extra line-cards and configuring a "virtual router."

Design Considerations for an IP backbone Router

This section attempts to highlight relevant requirements and explore a "first cut" architectural partitioning for a backbone routing device that is capable of line-rate performance at 40 Gbps. The first section involves specifying the functional requirements at a system level, while the second explores architectural and component level implications.

Functional Requirements

Bandwidth / Network Interface Requirements

Port Configuration Options
- 16 OC-48c (Packet over SONET
- 64 OC-12c (Packet over SONET)
- 256 OC-3c (Packet over SONET)
- 4 Gig Ethernet

Route / Flow Requirements
- 500K Routes
- 500K Flows
- Full DiffServ / MPLS flow classification with Multi label capabilities

Hard Performance guarantees
- Deterministic Forwarding Latencies
- Non blocking Sustained wire-speed operation for min size packets (40 bytes) for 16 OC-48c links
- Full wire-speed Multicast

Quality of Service

Admission Control
- Committed Access Rates for Flows based on DSCP / MPLS: The router should be configurable to enforce Service Level Agreements based on traffic flows that are derived from either a DiffServ Label, or an MPLS tag. The user should be able to program in a maximum "flow rate," which when exceeded will result in discarding all out-of profile packets.

Traffic Shaping

• Delivery Priorities / Drop Classes: The router should be able to support a minimum of 8 delivery priorities and 2 drop classes per channelized physical interface.
• Weighted Fair Queuing: The user should be able to configure the rate at which packets egress onto the link by providing a "weight" to each of the egress queues which may be derived from the content of the queue and will include fairness based on packet size.
• Weighted Round Robin Queuing: The round robin scheduler assigns each queue a fixed slot time which is equal across all queues by default. The user should be able to assign weights to each slot, thereby skewing the amount of time the scheduler grants to each queue.
• Full Wire-Speed Multicast support

Congestion Control
• Weighted Random Early Detection: User configurable thresholds for congestion and packet drop profiles.

Physical Dimension Constraints
• The design goal is to have the entire system fit into a 19 inch rack with a maximum height of 9 rack units.

Software
• Support for BGP-4, OSPF and RIP as well as constraint based MPLS routing.

Design Partitioning

Driving Forces and their Implications

Scalability, performance and power requirements drive system partitioning. A distributed packet forwarding architecture clearly lends itself to a more scalable system. The distributed architecture allows for building out of a maximum capacity chassis and backplane, and de-couples the scaling of the Network layer from the switch fabric. To summarize, we have taken a "divide and conquer" approach to solving system performance and scaling issues by:

• Reducing the complexity of the switch fabric and making it an ultrafast highly integrated dedicated data transport layer
• Building a chassis with an optical backplane (fiber) that can scale as high as 20 Gbps per link.
• Decoupling network layer from switch fabric allowing for maximum flexibility in scaling each independently.

The power and space requirements dictate a minimum number of components on each line-card. Power constraints dictate efficient utilization of silicon. Generalized network processing components may not be able to provide the optimal power / functionality point required for function specific systems. The solution requires function specific ASICs that efficiently implement Network Layer and QoS/Traffic Engineering functions. It is important to note that the tradeoffs associated with distributed routing include complexity in the control and management planes. The maintenance and updating of distributed packet forwarding intelligence may necessitate a more powerful processor and will result in more complex protocol structures. This, however is a small price to pay for scalable redundant line-rate performance. As mentioned earlier, the clear separation of the software routing control plane from the packet

forwarding path is key to the realization of line-rate performance. Figure 7 indicates the architecture of a typical line-card and switch fabric card for the proposed system, and delineates relevant component level requirements for optimal realization.

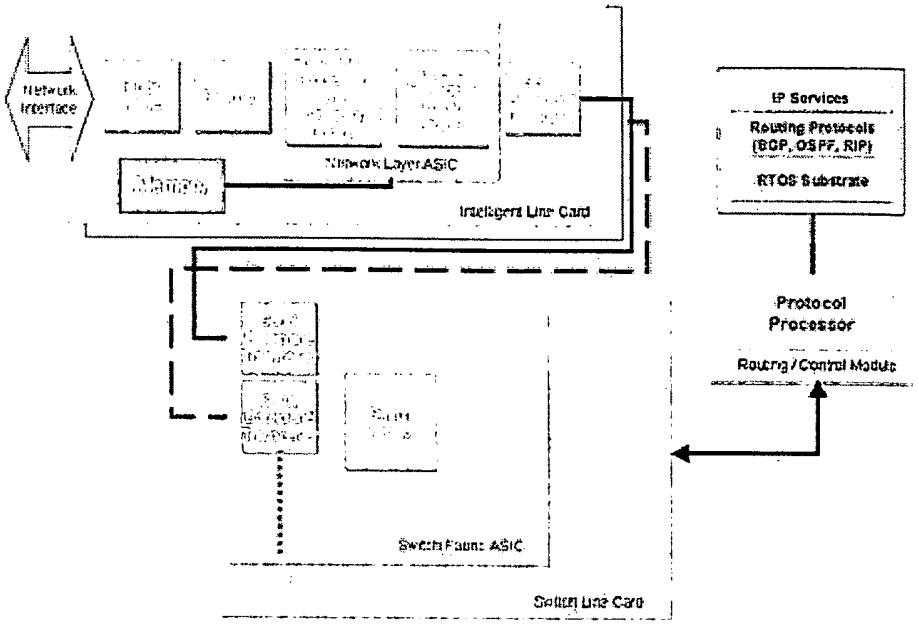
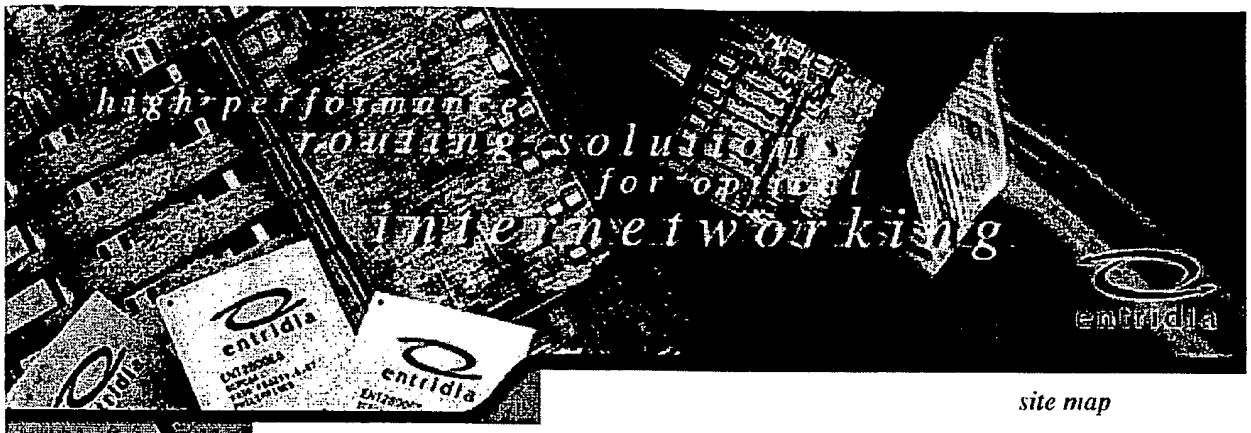


Figure 7. Concept Router Design and Implementation
(Click on image for a larger version)

Conclusions

It is clear that Internet growth coupled with service-laden traffic requires a new breed of Internet protocol-specific routers. These backbone devices require ultra-high connection densities, low power, and small form factors. Additionally, they require user configurable sophisticated traffic Engineering mechanisms that are MPLS and DiffServ based and tightly controlled deterministic packet forwarding performance. The basic set of requirements for a scalable Internet backbone device requires separation of the control plane from the packet forwarding plane; and distributing the packet forwarding function on a per line-card basis. This approach allows the switch fabric to scale at the same pace as the optical transport layer, enabling an ultra-high performance architecture that is only limited by the fundamental physical data-transport layer.

Last updated - May 25, 2000.



site map

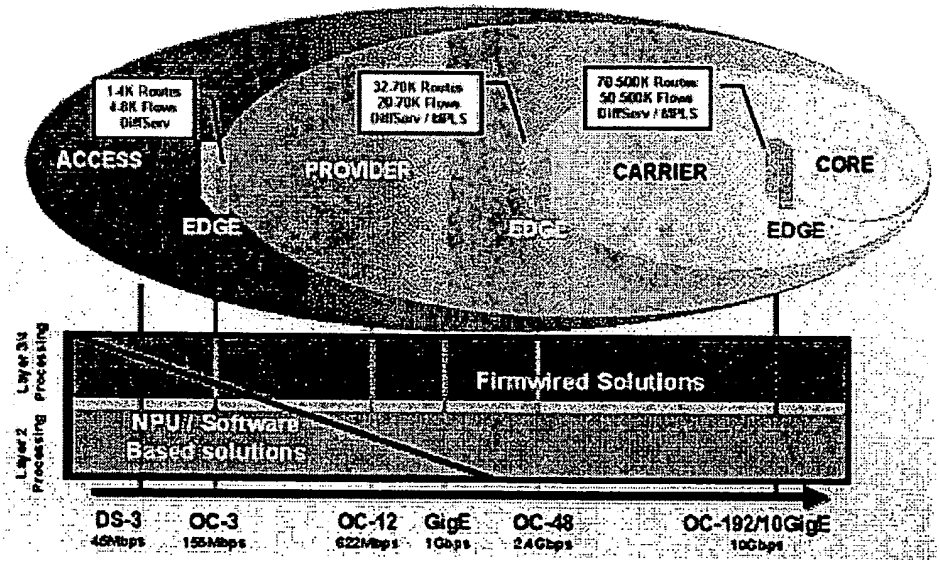
home
about us
technology
products
support
jobs
news & events
partners
contact us

technology 101

Gigabit IP Routing - Implementation Realities

GigE and 10GigE are being viewed as serious contenders for high performance link layer transport technologies in IP line aggregation equipment. Silicon based firmwired IP routing is the key to realizing the key edge functions in a heavily oversubscribed high-performance service based network.

Internetscape - Performance View



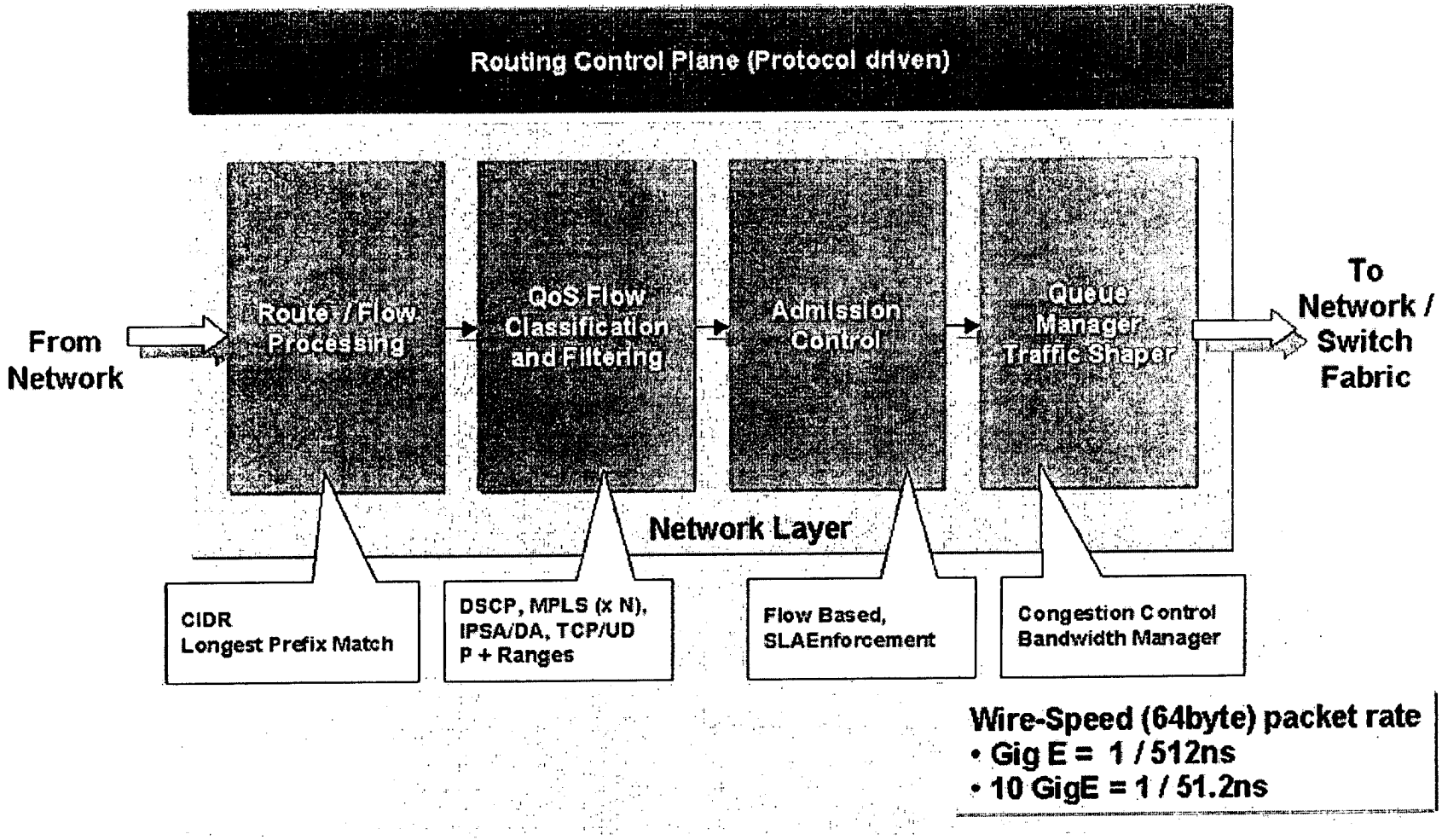
(Click on image for a larger version)

Requirements for a Converged Internet

Service Level Requirements

- **Guaranteed wire-speed routing for services**

  - Voice and Video cannot tolerate non-deterministic delays and jitter through routers


Routing Control Plane (Protocol driven)
From Network
Route / Flow Processing
QoS Flow Classification and Filtering
Admission Control
Queue Manager Traffic Shaper
To Network / Switch Fabric
Network Layer
CIDR Longest Prefix Match
DSCP, MPLS (x N), IPSA/DA, TCP/UDP + Ranges
Flow Based, SLAEnforcement
Congestion Control Bandwidth Manager
Wire-Speed (64byte) packet rate
• Gig E = 1 / 512ns
• 10 GigE = 1 / 51.2ns

- **Per-packet classification**

  - Examine and dynamically classify every packet using an arbitrary set of filtering rules to implement wide range of policies

- **Flow Maintenance and QoS**

  - Maintain and implement bandwidth provisioning information for e-services

- **Policy driven congestion recovery**

  - Support congestion control mechanisms established by higher level policies

System Level Requirements

- **Maximize the number of connections per rack foot**

  - CO space is a premium commodity

- **Minimize the cost per connection**

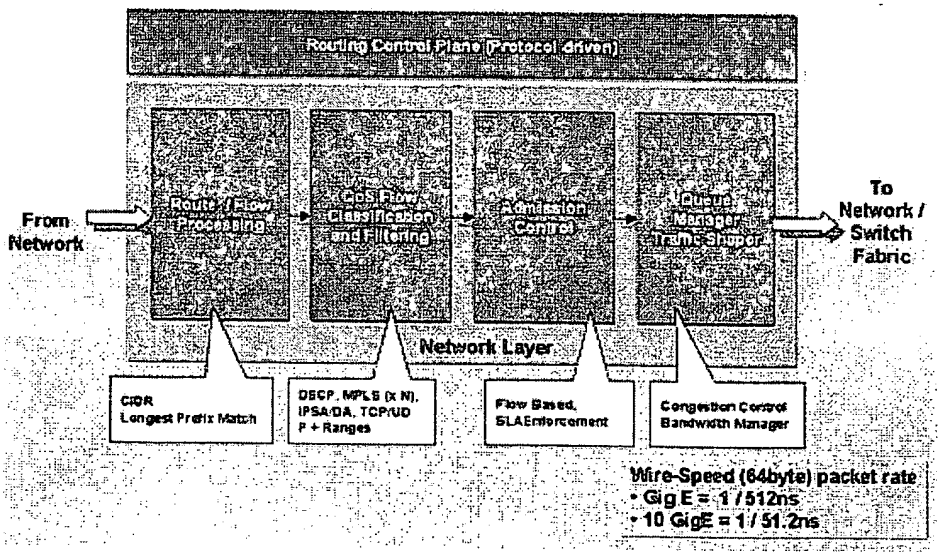
  - Connectivity is no longer a billable item, services become billables

- **Minimize the power figure per rack foot**

  - CO / Carriers are limited by their infrastructural power / heat management requirements

New Metrics for Carrier Gear
Routed Gigabits / (BTU x $ x rack ft.)

Wire-Speed QoS Enabled IP Routing
A Qualitative Analysis



(Click on image for a larger version)

Wire-Speed QoS Enabled IP Routing
Key Implementation Challenges

- **Access to External Packet Memory**

  - SDRAM becomes less viable because of its low efficiency and seriously impacts the ability to maintain wire-speed performance for back to back

min. size packets and multicast flows.

- **Access to Route / Flow Memory**

  - Algorithmic approaches have scalability issues and are corner optimized resulting in unbounded jitter and latency.

  - CAM's are a viable option, but may require architectural changes to become more efficient in range based filtering mechanisms.

- **Admission Control Mechanisms**

  - Any policy enforcement mechanism must be able to "sense" violations at line-rate.

- **Switch Fabric Scaling**

  - Architectures that concentrate the intelligence in the Switch Fabric will face serious scalability issues as well as power issues.

Conclusions

- **Configurable ASIC's are Key to Optimal Router Design. These fundamental building blocks should offer:**

  - highest integration (port density / functionality)

  - lowest possible power

  - easy software integration path (no forklift upgrade!)

  - optimal access to external agents (memory / processors / etc.)

- **Backplane Switch Fabrics should be as scalable as Physical media technology, and should focus on integrating serial connectivity blocks. The Backplane Switch should be optimized for data transport -- not network layer functions.**

Last updated - May 25, 2000.

CHAPTER 1

Planning for Quality of Service

The Cisco QoS Policy Manager is a full-featured QoS policy system that simplifies the complexity of configuring and deploying QoS policies for enterprise networks. It provides users with the ability to enable differentiated services, thereby providing better service for selected network traffic. This improves control over network resources and improves the cost efficiency of Wan connections. Also, it automates the task of QoS policy deployment. Relying on IP Precedence to enforce QoS policy end-to-end, QoS Policy Manager enables customers to quickly apply a mix of QoS policy objectives that expedite the handling of mission-critical applications.

Introduction

The Cisco QoS Policy Manager allows the user to define rules-based policies that controls QoS application traffic across the network. The QoS Policy Manager can be used to set policies on devices that will control the bandwidth when there is bandwidth contention, otherwise data flow is uninhibited. Using a high level graphical user interface, the user can define a QoS policy for multiple devices and interfaces, validate a policy prior to deploying it to the network, and monitor the deployment of QoS policies to network devices. QoS Policy Manager provides a network view of QoS policies deployed in the network and maintains an audit trail of all policy distributions.

Features at a Glance

- **Centralized Policy Control:** An easy to use graphical user interface for creating, modifying and deploying QoS policies.
- **Differentiated Service for Application Traffic:** Tools to simplify configuration of IP Precedence traffic classification used by Cisco devices to expedite high priority application traffic.
- **QoS Domain Configuration:** User defined groups of device interfaces upon which you can deploy a QoS policy. This allows you to easily enable a QoS policy across a domain or multiple devices.
- **Comprehensive QoS Feature Support:** Extensive congestion management and congestion avoidance services that can be deployed to improve network performance.
- **Reliable Policy Deployment:** Policy validation checking, preview of configuration file changes, partial ACL updates, and job control facilities allow you to reliably deploy QoS policies.
- **Web-Based Reporting:** Web-based reports allow you to quickly view and analyze QoS policy deployment.
- **Broad Device and IOS Release Support:** QoS Policy Manager device support includes the 2500, 3600, 4000, 4500, 4700, 7200 and 7500 series routers, the Router Switch Module (RSM) for the Catalyst 5000, the 8510 and Local Director v3.1.1. IOS Releases supported include 11.1, 11.2, 11.3, 12.0, and 11.1cc.
- **Resource Manager Essentials Integration:** Device inventory import from Resource Manager Essentials 2.0 that shortens the setup time for devices targeted for policy enforcement.

GUI Components

The QoS Policy Manager has two GUI components and a back-end process.

- **Policy Manager**—a GUI component that allows the user to define the devices managed by the policy system. Using the Policy Manager, the user defines QoS policies that can be applied to a single or to multiple devices and interfaces. The Policy Manager provides web-based reports of all QoS policies deployed in the network.
- **Distribution Manager**—A GUI component that allows the user to deploy saved policies to target devices. This tool maintains a history of all policy deployments including a detailed log of device configuration changes.

Back-End Process

QoS Manager—A back-end process that is the heart of the system. QoS Manager communicates with both the Policy Manager and the Distribution Manager, stores the policy database and configures the network devices.

The system has a distributed architecture. The GUI components may or may not be installed on the same computer as the QoS Manager. A Complete installation places all the components on one computer. A remote installation places the GUI components on a different computer from the QoS Manager. The remote installation allows the user to make, change, and distribute policies from a remote location.

Effective use of Quality of Service (QoS) capabilities requires careful planning. Before you deploy QoS to your network, carefully consider the types of applications used in your network and which QoS techniques might improve the performance of those applications.

These topics can help you plan for QoS deployment:

- What Is Quality of Service?
- How Does QoS Policy Manager Help Quality of Service?
- Planning for QoS Deployment
- More Information About Quality of Service

What Is Quality of Service?

Quality of Service (QoS) is a set of capabilities that allow you to create differentiated services for network traffic, thereby providing better service for selected network traffic. For example, with QoS, you can increase bandwidth for critical traffic, limit bandwidth for non-critical traffic, and provide consistent network response, among other things. This allows you to use expensive network connections more efficiently, and to establish service level agreements with customers of the network.

To implement QoS, you define QoS properties and policies on device interfaces. The policies can differentiate traffic based on its source, destination, or type. For example, you can recognize traffic based on the network host, port, protocol, or even IP precedence and TOS values in the packets.

QoS Policy Manager helps to provide end-to-end QoS service across a complete network.

Figure 1-1 QoS Policy Manager across a complete network

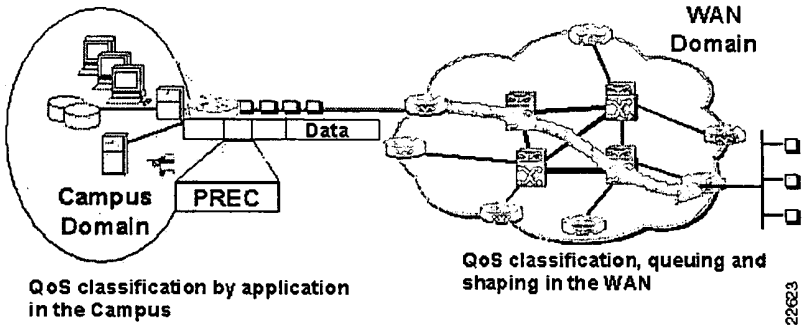


Figure 1-1 displays the conceptual diagram of a typical network. QoS Policy Manager normally classifies data packets in the Campus Domain (RSM for Catalyst 5000, 8510 and LocalDirector). The classification is what the WAN devices (7200, 7500, 4700, 4500, 4000, 3600, and 2500) first examine at the input interface. The packet information is then compared to the QoS policies for the output interface. Based upon the QoS policies, the data packet is queued for output. The QoS policies that determine output queuing may include shaping policies or limiting policies.

Table 1-1 Cisco QoS Policy Manager Features, Uses, and Benefits

Feature	Description	Use/Benefit
	QoS Services	
Traffic Classification	IP address and ports that are referenced for IP precedence and CAR excess bandwidth precedence setting.	Efficiently classifies packets into traffic classes to provide powerful, flexible traffic prioritization.
Queue Management	Enforces class of service policies using priority, custom, or weighted fair queuing.	Eliminates classifying traffic explicitly at each WAN interface in the backbone network.
Congestion Control	CAR limiting, shaping, and FRTS policy enforcement.	Controls congestion on critical links by configuring rate enforced on outbound traffic.
Congestion Avoidance	WRED	Avoids conditions that degrade throughput.

QoS primarily comes into play when the amount of traffic through an interface is greater than the interface's bandwidth. When the traffic through an interface exceeds the bandwidth, packets form a *queue* from which the device selects the next packet to send. The selection is based upon preset criteria. The following topics describe how QoS manages traffic based upon selectable criteria.

- What Devices and IOS Software Releases Are Supported?
- Understanding Policy Implementation Sequence on an Interface
- Traffic Coloring Techniques
- Queuing Techniques for Congestion Management on Outbound Traffic
- Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth
- Queuing Techniques for Congestion Avoidance on Outbound Traffic

What Devices and IOS Software Releases Are Supported?

Table 1-2 describes the devices and IOS software releases that QoS Policy Manager supports, and the QoS techniques you can use on the supported platforms. QoS Policy Manager, version 1.0 supports up to 200 devices in a network.

Table 1-2 Supported Devices, QoS Techniques, and IOS Software Releases

Quality of Service Technique	Cisco Systems Device	IOS Software Release				
		11.1	11.2	11.3	12.0	11.1(cc)
Priority queuing (PQ), custom queuing (CQ)	7500, 7200	X	X	X	X	X
	RSM, 4700, 4500, 3600	X	X	X	X	
	2500, 4000	X	X	X	X	
Weighted random early detection (WRED)	7500, 7200, RSM		X	X	X	
	4700, 4500, 4000		X	X	X	
	3600, 2500		X	X	X	
Weighted fair queuing (WFQ)	7500, 7200		X	X	X	X
	4700, 4500, 4000, RSM		X	X	X	
	3600, 2500		X	X	X	
Policy Based Routing (PBR)	7500, 7200, RSM		X	X	These devices use CAR coloring	
	4700, 4500		X	X		
	3600		X	X		
	4000, 2500		X	X	X	
Generic (GTS)	7500, 7200, RSM		X	X	X	
	4700, 4500, 4000		X	X	X	
	3600, 2500		X	X	X	

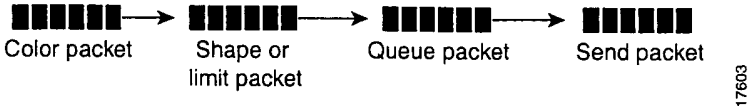
Table 1-2 Supported Devices, QoS Techniques, and IOS Software Releases (continued)

Quality of Service Technique	Cisco Systems Device	IOS Software Release				
		11.1	11.2	11.3	12.0	11.1(cc)
Frame Relay traffic shaping (FRTS)	7500, 7200, RSM		X	X	X	
	4700, 4500		X	X	X	
	3600		X	X	X	
	4000		X	X	X	
	2500		X	X	X	
Committed Access Rate (CAR) Classification	7500, 7200				X	X
	RSM, 4700, 4500				X	
	4000, 2500					
Committed Access Rate (CAR) Rate Limit	7500, 7200				X	X
	RSM, 4700, 4500				X	
	4000, 2500					
Weighted Round Robin (WRR)	8510				X	
Packet Classification	Local Director 3.1.1					

Understanding Policy Implementation Sequence on an Interface

Understanding the sequence in which policies are implemented by an interface can help you define meaningful policies that implement your traffic management requirements. Figure 1-2 shows the sequence a packet follows when it reaches an interface.

Figure 1-2 Sequenced Used to Implement Interface Policies on a Packet



When a packet reaches an interface, the interface acts upon the packet in the following sequence:

- Normally coloring policies are applied first and at the input interface. For most interfaces, the first policy match found ends the search through the defined policies: that is, each policy is compared to the packet, and when a match is found, the policy is applied and no other policies are considered. There may, or may not, be a coloring policy.
- Next shaping policies or limiting policies are applied. Shaping policies and limiting policies are applied to shape or limit output bandwidth. Shaping policies and limiting policies are not interchangeable. A shaping policy constricts the flow, but does not drop packets when it reaches the rate limit, unless the flow rate causes the volume of packets to exceed the device buffer. This allows for spikes in traffic flow. A limiting policy drops all packets that exceed the rate limit. There may, or may not, be a shaping policy or a limiting policy.
- Queuing the packet. After any policies are applied to a packet, the packet is queued to leave the interface. At this point, if a coloring policy was applied to a packet, it can affect how the packet is queued. If shaping policies or limiting policies were applied, it affects the bandwidth available to the packet.
- Send the packet to the designated IP address.

  With some IOS software versions and device models, you can define a policy so that subsequent policies are considered after a match is found. In these cases, you can color a packet in one policy at the input interface, and apply a limiting policy to the same packet, perhaps by keying on the packet's color. Refer to Table 1-2 to see which combinations support Committed Access Rate (CAR) limiting or CAR classification. Normally, you should apply a coloring policy prior to applying a limiting policy. However, in some CAR cases a limiting policy can be applied at the input interface before applying a coloring policy.

Traffic Coloring Techniques

Some interface QoS properties recognize a packet's relative importance by examining the IP precedence field in the packets header. Changing the IP precedence value is changing the packets *color*. Coloring is a property that when attached to a packet affects the way the packet is handled on its entire path through the network.

By changing a packet's color on an inbound interface, you can affect how your devices handle the packet. Interfaces that use WFQ or WRED automatically recognize and use the IP precedence value. Priority queuing and custom queuing do not automatically consider the IP precendence (coloring) of a packet. Therefore, to have coloring affect how the packet gets prioritized on a priority queue or custom queue interface, you must create an additional policy on the outbound interface that recognizes the traffic and places it in the appropriate queue (in addition to creating a coloring policy on the inbound interface). The same applies for traffic shaping and traffic limiting.

With IOS 11.1cc and IOS 12.0, coloring policies are implemented using Committed Access Rate (CAR). CAR allows the user to set policies for coloring and priority queuing, custom queuing, shaping or limiting, to be implemented on the same interface. (See Chapter 5, "Defining a Coloring Action.") By using complex coloring policies, you can create a policy that colors a packet, then tells the IOS to examine other policies set on that interface. For example, the first policy would color the packet, then in a subsequent policy, you can key on the packet's color to apply a shaping, limiting, priority queuing or custom queuing policies.

Because IOS software applies QoS coloring policies on the input interface before queuing a packet, the coloring policy you set can affect how that packet is queued on the interface.

Interface QoS Property Requirements for Colored Traffic

You can define traffic coloring policies on any type of interface. WFQ and WRED automatically consider the packet's color when queuing the packet.

For any other queuing policies, shaping policies or limiting policies you must first define a coloring policy for the inbound interface. The color key may then be examined by IOS to determine other policies for policy based routing (PBR).

In IOS software versions 11.1cc and 12.0 committed access rate (CAR) allows the user to set polices that will apply the coloring policy on the inbound or outbound interfaces. IOS then examines other policies for queuing and bandwidth control.

Related Topics

- Defining a Coloring Action (Chapter 5)

- Lesson 3: Creating a Simple Policy for Managing Web Traffic on One Router (Chapter 3)

Queuing Techniques for Congestion Management on Outbound Traffic

You can set a queuing technique on a device's interface to manage how packets are queued to be sent through the interface. The technique you choose determines whether the traffic coloring characteristics of the packet are used or ignored.

These queuing techniques are primarily used for managing traffic congestion on an interface, that is, they determine the priority in which to send packets when there is more data than can be sent immediately:

- First In, First Out (FIFO) Queuing: Basic Store and Forward
- Priority Queuing (PQ): Basic Traffic Prioritization
- Custom Queuing (CQ): Advanced Traffic Prioritization
- Weighted Fair Queuing (WFQ): Intelligent Traffic Prioritization
- Weighted Round Robin (WRR): Traffic Taking Turns

First In, First Out (FIFO) Queuing: Basic Store and Forward

FIFO queuing is the basic queuing technique. In FIFO queuing, packets are queued on a first come, first served basis: if packet A arrives at the interface before packet B, packet A leaves the interface before packet B. This is true even if packet B has a higher IP precedence than packet A: FIFO queuing ignores packet characteristics.

FIFO queuing works well on uncongested high-capacity interfaces that have minimal delay, or when you do not want to differentiate services for packets traveling through the device.

The disadvantage with FIFO queuing is that when a station starts a file transfer, it can consume all the bandwidth of a link to the detriment of interactive sessions. The phenomenon is referred to as a *packet train* because one source sends a "train" of packets to its destination and packets from other stations get caught behind the train.

Policy Requirements for FIFO Queuing Interfaces

There are no specific requirements for creating policies on FIFO interfaces. You do not have to define any policies on these interfaces.

However, you can create traffic shaping policies or traffic limiting policies on FIFO interfaces to set the rate limit on the bandwidth available to selected traffic. You can color the traffic on a FIFO interface but it cannot be used to affect FIFO queuing.

FIFO's Relationship to Traffic Coloring

FIFO queuing treats all packets the same: whichever packet gets to the interface first is the first to go through the interface. Traffic shaping and traffic limiting policy statements can affect the bandwidth available to a packet based on its color, but FIFO does not use the coloring value.

Related Topics

- Traffic Coloring Techniques
- Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth

Priority Queuing (PQ): Basic Traffic Prioritization

Priority queuing is a rigid traffic prioritization scheme: if packet A has a higher priority than packet B, packet A always goes through the interface before packet B.

When you define an interface's QoS property as priority queuing, four queues are automatically created on the interface: high, medium, normal, and low. Packets are placed in these queues based on previously defined priorities and policies. Unclassified packets are placed in the normal queue.

The disadvantage of priority queuing is that the higher queue is given absolute precedence over lower queues. For example, packets in the low queue are only sent when the high, medium, and normal queues are completely empty. If a queue is always full, the lower-priority queues are never serviced. They fill up and packets are lost. Thus, one particular kind of network traffic can come to dominate a priority queuing interface.

An effective use of priority queuing would be for placing time-critical but low-bandwidth traffic in the high queue. This ensures that this traffic is transmitted immediately, but because of the low-bandwidth requirement, lower queues are unlikely to be starved.

Policy Requirements for Priority Queuing Interfaces

In order for packets to be classified on a priority queuing interface, you must create policies on that interface. These policies need to filter traffic into one of the four priority queues. Any traffic that is not filtered into a queue is placed in the normal queue.

You can also create traffic shaping policies or traffic limiting policies to define an upper range on the bandwidth allocated to selected traffic.

Priority Queuing's Relationship to Traffic Coloring

Priority queuing interfaces do not automatically consider the IP precedence settings of a packet. If you create traffic coloring policies on inbound interfaces (see "Traffic Coloring Techniques"), and you want the coloring to affect the priority queue, you must create a policy on the priority queuing outbound interface that recognizes the color value and places the packet in the desired queue.

Note With IOS versions 11.1cc and 12.0 you can color traffic on inbound or outbound interfaces.

Related Topics

- Defining a Priority Queuing Action for Outbound Traffic (Chapter 5)
- Traffic Coloring Techniques
- Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth

Custom Queuing (CQ): Advanced Traffic Prioritization

Custom queuing is a flexible traffic prioritization scheme that allocates a minimum bandwidth to specified types of traffic. You can create up to 16 of these custom queues.

For custom queue interfaces, the device services the queues in a round-robin fashion, sending out packets from a queue until the byte count on the queue is met, then moving on to the next queue. This ensures that no queue gets starved, in comparison to priority queuing.

The disadvantage of custom queuing is that, like priority queuing, you must create policy statements on the interface to classify the traffic to the queues.

An effective use of custom queuing would be to guarantee bandwidth to a few critical applications to ensure reliable application performance.

Policy Requirements for Custom Queuing Interfaces

In order for packets to be classified on a custom queuing interface, you must create custom queuing policies on that interface. These policies need to specify a ratio, or percentage, of the bandwidth on the interface that should be allocated to the queue for the filtered traffic. A queue can be as small as 5%, or as large as 95%, in increments of 5%. The total bandwidth allocation for all policy statements defined on a custom queuing interface cannot exceed 95% (QoS Policy Manager ensures that you do not exceed 95%). Any bandwidth not allocated by a specific policy statement is available to the traffic that does not satisfy the filters in the policy statements.

QoS Policy Manager uses the ratio in these policies, along with the packet size specified when you define an interface as a custom queue, to determine the byte count of each queue.

The queues you define constitute a minimum bandwidth allocation for the specified flow. If more bandwidth is available on the interface due to a light load, a queue can use the extra bandwidth. This is handled dynamically by the device.

All packets that have not been classified for custom queuing are placed in the default queue.

You can also create traffic shaping policies or traffic limiting policies to define an upper range on the bandwidth allocated to selected traffic. Thus, the custom queue defines a minimum bandwidth, and the shaping policy or limiting policy defines an upper limit. When defining the bandwidth upper limit, the shaping or limiting policy must be executed before the custom queue policy, and it must filter the same traffic as the custom queue (or a subset of the same traffic).

Custom Queuing's Relationship to Traffic Coloring

Custom queuing interfaces do not automatically consider the IP precedence settings of a packet. If you create traffic coloring policies on inbound interfaces (see "Traffic Coloring Techniques"), and you want the coloring to affect the custom queue, you must create a policy on the custom queuing outbound interface that recognizes the color value and places the packet in the desired queue.

Note With IOS versions 11.1cc and 12.0 you can color traffic on inbound or outbound interfaces.

Related Topics

- Defining a Custom Queuing Action for Outbound Traffic (Chapter 5)
- Lesson 4: Coloring Enterprise Resource Planning (ERP) Traffic on a Group of Devices (Chapter 3)
- Traffic Coloring Techniques
- Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth

Weighted Fair Queuing (WFQ): Intelligent Traffic Prioritization

Weighted fair queuing acknowledges and uses a packet's priority without starving low-priority packets for bandwidth. Weighted fair queuing divides packets into two classes: interactive traffic is placed at the front of the queue to reduce response time; non-interactive traffic shares the remaining bandwidth proportionately.

Because interactive traffic is typically low-bandwidth, its higher priority does not starve the remaining traffic. A complex algorithm is used to determine the amount of bandwidth assigned to each traffic flow. IP precedence is considered when making this determination.

Weighted fair queuing is very efficient, and requires little configuration.

Policy Requirements for Weighted Fair Queuing Interfaces

Weighted fair queuing interfaces automatically create queues for each traffic flow. No specific policies are needed.

However, you can also create traffic shaping policies or traffic limiting policies to affect how selected traffic is handled on the interface. A shaping policy or a limiting policy can control the bandwidth available to the selected traffic, whereas a coloring policy can change the relative importance of a packet and thus change how the interface queues the traffic.

Weighted Fair Queuing's Relationship to Traffic Coloring

Weighted fair queuing is sensitive to the IP precedence settings in the packets. Weighted fair queuing automatically prioritizes the packets without the need for you to create policies on the weighted fair queuing interfaces. However, if you do create a coloring policy on the weighted fair queuing interface, it will affect how the selected traffic is queued. Weighted fair queueing can improve network performance without traffic coloring policies.

When using coloring for weighted fair queuing, or WRED, you should be aware that traffic starting devices can set precedence fields in their outgoing packets. Thus, it is important to set IP precedence on all network access points in order to avoid unintended traffic receiving high priority.

Note With IOS versions 11.1cc and 12.0 you can color traffic on inbound or outbound interfaces.

Related Topics

- Lesson 4: Coloring Enterprise Resource Planning (ERP) Traffic on a Group of Devices (Chapter 3)
- Traffic Coloring Techniques
- Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth

Weighted Round Robin (WRR): Traffic Taking Turns

In Weighted Round Robin (WRR) there are four queues for each interface. Incoming traffic is placed in one of the four queues based upon precedence. The system gathers IP precedence information from the service type field of the IP header. For an incoming IP packet, the first two (most significant) bits of the service type field determine the priority. The system recognizes the four QoS classes as summarized in Table 1-3.

Table 1-3 QoS Queue Assignment Precedence

Service Type Field Value	Priority
000	00
001	00
010	01
011	01
100	10
101	10
110	11
111	11

Bandwidth is not explicitly reserved for these four queues. Each queue is assigned a different weighted round-robin (WRR) scheduling weight, which determines the way they share the interface bandwidth. The WRR is configurable. You can assign a different WRR weight for each queue. The higher the WRR weight, the higher the effective band width for that particular queue.

You can determine the effective bandwidth (in Mbps) for a particular queue using the following formula:

(W/S) X B = n Mbps

where

W is the WRR weight of the specified queue

S is the sum of the weight of all active queues on the outgoing interface

B is the available bandwidth in Mbps

For example, if W is 4, S is 15, and B is 100 Mbps bandwidth, the equation is:

(4/15) X 100 = 26 Mbps

Thus the effective bandwidth for this queue is 26 Mbps.

The weight for any queue is from 0 - 15, however, the sum of the WRR weight for all four queues on an interface is 15. If the sum of the WRR weights exceed 15 you are likely to exceed the bandwidth.

Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth

You can create traffic shaping policies or traffic limiting policies on a device's interface to manage how much of the interface's bandwidth should be allocated to a specific traffic flow. You can set your policies based on a variety of traffic characteristics, including the type of traffic, its source, its destination, and its IP precedence settings (traffic coloring). Shaping differs from limiting in that shaping attempts to throttle traffic when it reaches the rate limits. The router buffers some of the traffic bursts. Only when the buffer fills are packets dropped. Whereas, limiting policies do not drop packets until rate limits are reached, then drop all packets that exceed the rate limit.

Unlike queuing techniques, which are part of an interface's characteristics, generic traffic shaping or traffic limiting is done through policies, which are defined in access control lists (ACLs). (Frame relay traffic shaping, FRTS, is defined in interface characteristics.) Queuing techniques only affect traffic when an interface is congested, or in the case of WRED, when traffic exceeds a certain threshold. With traffic shaping policies, flows are affected even during times of little congestion.

You can use these types of traffic shaping policies:

- Generic Traffic Shaping (GTS): Controlling Traffic on Non-Frame Relay Interfaces

- Committed Access Rate (CAR): Controlling Traffic at a Committed Rate
- Frame-Relay Traffic Shaping (FRTS): Controlling Traffic on Frame Relay Interfaces and Subinterfaces
- Limiting Bandwidth: Limiting Bandwidth and Optionally Coloring Traffic

Generic Traffic Shaping (GTS): Controlling Traffic on Non-Frame Relay Interfaces

Generic traffic shaping lets you set a bandwidth limit for specific types of traffic. For example, you can create a policy that limits web traffic to 200 KB/sec. This puts a cap on the bandwidth available to that traffic, ensuring that the remainder of the interfaces bandwidth is available to other kinds of traffic. In this example, if web traffic does not fill 200 KB/sec, other kinds of traffic can use the unused bandwidth.

With generic traffic shaping, you can define a buffer to accommodate traffic bursts, so that packets are not immediately dropped once the limit is reached. If you do not define a buffer, once the limit is reached, packets are dropped.

Interface QoS Property Requirements for Generic Traffic Shaping

You can define generic traffic shaping policies on any type of interface except those that use frame relay traffic shaping (FRTS).

For custom queuing interfaces, you can create a shaping policy to form an upper limit for the bandwidth available to the selected traffic, and have the interface also apply a custom queuing policy to form a lower bandwidth limit for the traffic. However, this is only available on selected combinations of IOS software versions and device models (see Table 1-2 to see which combinations support Committed Access Rate (CAR) limiting or CAR classification).

Related Topics

- Defining a Shaping Policy Action for Outbound Traffic (Chapter 5)
- Lesson 5: Limiting the Bandwidth Available to FTP Traffic (Chapter 3)

Committed Access Rate (CAR): Controlling Traffic at a Committed Rate

With IOS 11.1cc and IOS 12.0, coloring policies are implemented using Committed Access Rate (CAR). CAR allows the user to set policies for coloring and priority queuing, custom queuing, shaping or limiting, to be implemented on the same interface. (See Chapter 5, "Defining a Coloring Action.") By using complex coloring policies, you can create a policy that colors a packet, then tells the IOS to examine other policies set on that interface. For example, the first policy would color the packet, then in a subsequent policy, you can key on the packet's color to apply a shaping, limiting, priority queuing or custom queuing policies.

For custom queuing interfaces, you can create a limiting policy to form an upper limit for the bandwidth available to the selected traffic, and have the interface also apply a custom queuing policy to form a lower bandwidth limit for the traffic.

CAR advanced settings allows you to set rate, burst size, exceeded burst size, direction, conformation priority, and exceeded priority. This allows you to set both coloring and limiting on both input and output interfaces.

Interface QoS Property Requirements for CAR Traffic Shaping

You can define traffic coloring policies on any type of interface. WFQ and WRED automatically consider the packet's color when queuing the packet.

In IOS software versions 11.1cc and 12.0 committed access rate (CAR) allows the user to set polices that will apply the coloring policy on the inbound or outbound interfaces. IOS then examines other policies for queuing and bandwidth control. For custom queuing interfaces, you can create a shaping policy to form an upper limit for the bandwidth available to the selected traffic, and have the interface also apply a custom queuing policy to form a lower bandwidth limit for the traffic.

Related Topics

- Defining a Shaping Policy Action for Outbound Traffic (Chapter 5)
- Lesson 5: Limiting the Bandwidth Available to FTP Traffic (Chapter 3)

Frame-Relay Traffic Shaping (FRTS): Controlling Traffic on Frame Relay Interfaces and Subinterfaces

Frame relay traffic shaping lets you specify an average bandwidth size for frame relay virtual circuits (VC). You can also define the bandwidth allocated for bursty traffic, and control whether the circuit responds to notifications from the network that the circuit is becoming congested. By using FRTS, you can define a minimum rate commitment for the virtual circuit, and accommodate the occasional need for greater bandwidth.

Each virtual circuit (VC) is identified by a data link connection identifier (DLCI) and provides access to another endpoint of the frame relay (FR) cloud. The VC can be either a switched VC (SVC) or permanent VC (PVC). In the SVCs a connection establishment is made each time data needs to be sent over the network (like ATM) and negotiation of parameters occurs. For PVC, the connection is there all the time and its parameters are defined when it is ordered from the FR carrier.

If the FR is a fully meshed network, the FR cloud is viewed by the router like a shared media subnet, similar to an Ethernet interface in which few other routers connect to it. There may be a few VCs on the interface connecting to the FR network, but it will not be divided into logical subinterfaces.

It is very common that the FR interface is not fully meshed but is a collection of virtual point-to-point links. In this case subinterfaces will be defined on the interface connecting to the FR network and therefore only one VC is defined on each of the subinterfaces.

QoS Policy Manager applies your FRTS definitions to all VCs defined on an interface or subinterface. You cannot treat multiple VCs on a single interface or subinterface differently.

Interface QoS Property Requirements for Frame-Relay Traffic Shaping

You can use FIFO, priority queuing, or custom queuing on frame relay subinterfaces. If you use priority queuing or custom queuing, you must create policies on the interfaces or subinterfaces that create the required queues. On a FR interface you can use WRED and weighted fair queuing (WFQ).

By creating custom or priority queues, you can further modify the automatic rate-limiting features of FRTS. Parameters you can control through QoS are Rate (CIR), burst size (BC), excess burst size (BE) and adaptive rate (response to BECN notifications).

You cannot create generic traffic shaping policies on an FRTS interface. You can create traffic coloring policies on the interface, however.

Note When applying FRTS to a subinterface, the system writes an enabling command to the parent interface.

Related Topics

- Priority Queuing (PQ): Basic Traffic Prioritization
- Custom Queuing (CQ): Advanced Traffic Prioritization

Limiting Bandwidth: Limiting Bandwidth and Optionally Coloring Traffic

Limiting lets you set a bandwidth limit for specific types of traffic. For example, you can create a policy that limits web traffic to 200 KB/sec. This puts a cap on the bandwidth available to that traffic, ensuring that the remainder of the interface's bandwidth is available to other kinds of traffic. In this example, if web traffic does not fill 200 KB/sec, other kinds of traffic can use the unused bandwidth.

Packets are dropped if traffic bursts exceed the limit.

Interface QoS Property Requirements for Rate Limited Traffic

You can define limiting policies on any type of interface.

For custom queuing interfaces, you can create a limiting policy to form an upper limit for the bandwidth available to the selected traffic, and have the interface also apply a custom queuing policy to form a lower bandwidth limit for the traffic. However, this is only available on selected combinations of IOS software versions and device models (see Table 1-2 to see which combinations support Committed Access Rate (CAR) limiting or CAR classification).

Related Topics

- Generic Traffic Shaping (GTS): Controlling Traffic on Non-Frame Relay Interfaces
- Custom Queuing (CQ): Advanced Traffic Prioritization
- Defining a Limiting Action (Chapter 5)

Queuing Techniques for Congestion Avoidance on Outbound Traffic

You can set a queuing technique on a device's interface to manage how packets are handled when an interface starts to be congested, in order to avoid the congestion. The queuing technique available for congestion avoidance is weighted random early detection (WRED).

With WRED, when traffic begins to exceed the interface's traffic thresholds, but before congestion occurs, the interface starts dropping packets from selected flows. If the dropped packets are TCP, the TCP source recognizes that packets are getting dropped, and lowers its transmission rate. The lowered transmission rate then reduces the traffic to the interface, thus avoiding congestion. Because TCP retransmits dropped packets, no actual data loss occurs.

To determine which packets to drop, WRED takes these things into account:

- RSVP flows are given precedence over non-RSVP flows, to ensure that time-critical packets are transmitted as required.
- The IP precedence of the packets. Packets with higher precedence are less likely to be dropped.
- The amount of bandwidth used by the traffic flow. Flows that use the most bandwidth are more likely to have packets dropped.
- The weight factor you have defined for the interface. The weight factor determines how frequently packets are dropped.

WRED chooses the packets to drop after considering these factors in combination, with the net result being that the highest priority and lowest bandwidth traffic is preserved.

WRED differs from standard random early detection (RED) in that RED ignores IP precedence, and instead drops packets from all traffic flows, not selecting low precedence or high bandwidth flows.

By selectively dropping packets before congestion occurs, WRED prevents an interface from getting flooded, necessitating a large number of dropped packets. This increases the overall bandwidth usage for the interface.

If you are using IOS software version 12.0 on a device with a versatile interface processor (VIP), when you configure an interface to use WRED it automatically uses distributed WRED. Distributed WRED takes advantage of the VIP.

The disadvantage of weighted random early detection is that only TCP/IP networks can benefit. Other protocols, such as UDP or Netware, do not respond to dropped packets by lowering their transmission rates, instead retransmitting the packets at the same rate. If you have a mixed network, WRED may not be the best choice for queuing traffic.

An effective use of weighted random early detection would be to avoid congestion on a predominantly TCP/IP network, one that has minimal UDP traffic and no significant traffic from other networking protocols. It is especially effective on core devices rather than edge devices, because the traffic coloring you perform on edge devices can then affect the WRED interfaces throughout the network.

Policy Requirements for Weighted Random Early Detection Interfaces

Weighted random early detection interfaces automatically drop packets for high-bandwidth, low priority traffic flows. No specific policies are needed.

However, you can also create traffic shaping policies or traffic limiting policies to affect how select traffic is handled on the interface. A shaping policy or a limiting policy can control the bandwidth available to the selected traffic, whereas a coloring policy can change the relative importance of a packet and thus change how the interface queues the traffic.

Weighted Random Early Detection's Relationship to Traffic Coloring

Weighted random early detection is sensitive to the IP precedence settings in the packets, so you can create policies on inbound interfaces on the device and have those policies implemented on the outbound interfaces that use weighted random early detection. Weighted random early detection automatically prioritizes the packets without the need for you to create policies on the weighted random early detection queuing interfaces, dropping packets with low priority before dropping high-priority packets. However, you do not need to create policies on the inbound interfaces that color traffic (see the "Traffic Coloring Techniques" section earlier in this chapter). If packets have the same IP precedence, weighted random early detection drops packets from the highest-bandwidth flows first. If you create a coloring policy on the weighted random early detection interface, it also affects how the selected traffic is queued.

Related Topics

- Traffic Coloring Techniques
- Traffic Shaping or Traffic Limiting Techniques for Controlling Bandwidth

How Does QoS Policy Manager Help Quality of Service?

QoS Policy Manager's GUI makes it easier for you to create Quality of Service policies, so that you do not have to manually connect to each of your devices and use IOS software commands to configure the policies.

QoS Policy Manager detects the QoS capabilities that are available on each of your devices, as defined by the device model, interface type, and the IOS software version running on the device. With QoS Policy Manager, you cannot select an unsupported QoS capability for a given interface. You can chose different QoS techniques for different interfaces, as appropriate, to implement your overall networking policies.

Table 1-4 Cisco QoS Policy Manager Features, Uses, and Benefits

Policy Definition and Validation		
Feature	**Description**	**Uses / Benefits**
Policy Abstraction	Graphical user interface translates policy statements into interface-specific configuration commands.	Eliminates the need to use different configuration commands and syntax to configure QoS features across diverse devices and IOS releases via the CLI.
Granular Policy Definition	Supports fine-grained policies that filter traffic based on IP address, source or destination port, protocol, IP precedence value, TOS value, host group and application service.	Flexibility defines highly differentiated services to enforce application and host traffic prioritization.

Table 1-4 Cisco QoS Policy Manager Features, Uses, and Benefits (continued)

Policy Definition and Validation

Feature	Description	Uses / Benefits
Policy Domain Management	Supports multi-device interface grouping for policy definition and deployment.	Speeds policy definition for multiple devices and ensures access list accuracy and consistency. Permits QoS deployment based on QoS policy domain.
IOS Version Transparency	Allows policy deployment for interface groups to the same software version or to different software versions, based on IOS compatibility.	Provides flexibility in policy definition and allows for software upgrades without affecting existing policy deployment.
Rules-Based Policy Filters	Policies are defined using a Boolean expression that specifies the QoS filter that is applied to IP packets.	Flexibly creates complex QoS policy conditions that combine applications and host systems.
Policy Validation	Prevents QoS policy deployment for policies not supported on the interface based on device model, IOS software version, and queuing technique.	Identifies conflict prior to policy deployment, reducing configuration errors. Ensures configuration file accuracy and consistency.
Policy Prioritization	Allows the administrator to change the priority of policies applied to each interface by specifying the order that route map statements are processed.	Ensures that policies automatically receive the required class of service.
Application Service Templates	Creates and maintains application service templates containing application port number, protocol, and TCP/UDP socket addresses.	Reduces the complexity of setting up application-based policies.

Table 1-4 Cisco QoS Policy Manager Features, Uses, and Benefits (continued)

Policy Definition and Validation (continued)

Feature	Description	Uses / Benefits
Host Group Classes	Creates and maintains host group classes containing IP address, DNS name or IP address and mask combination used during policy definition.	Scalable policy deployment to groups of users/hosts allowing for easy updates.
DNS Resolution	DNS names are resolved when devices are introduced into the policy system and during policy definition.	Eliminates manually resolving hosts names when IP address change occurs.
	Policy Deployment	
Policy Commands	Preview IOS configuration commands to be distributed to target devices.	Permits confirmation prior to committing policy changes to the network.
Incremental Configuration Update	Only distributes incremental changes in configuration commands.	Reduces time, bandwidth and processing required to propagate policy updates.
Job Control	Devices can be configured serially or in parallel. Provides ability to stop and resume a policy distribution at a later time.	Permits the halting of a QoS policy distribution allowing for timely corrective action.
Job Status and Logging	View job report showing job status and a detailed log of device operations.	Permits easy tracking of a multi-device policy distribution and all device operations.
Job History	Stores all job and device history information for easy viewing.	Provides a complete policy audit trail.
Web-based Reports	Generates HTML reports for QoS policies deployed by network, device and interface.	Easy-view QoS policy reports for status monitoring and troubleshooting.

Table 1-4 Cisco QoS Policy Manager Features, Uses, and Benefits (continued)

	Policy Server Architecture	
Feature	**Description**	**Uses / Benefits**
Device Inventory Import	QPM can import data in a variety of formats from Resource Manager Essentials 2.0 and Cisco resource Manager 1.1.	Supports integration with existing databases and speeds policy system configuration.
Device and Interface Attribute Knowledge	Detects and stores device model, IOS software version QoS feature, and interface type in the policy system knowledge base.	Reduces time required to manually input device information. Ensures accurate policy validation.

These topics cover the general way that interfaces apply policies to traffic, and the types of QoS capabilities you can implement with QoS Policy Manager:

- How Does QoS Policy Manager Help Quality of Service?
- Does QoS Policy Manager Ensure That My Policies Are Consistent?
- How Does QoS Policy Manager Support My Existing ACL Policies?

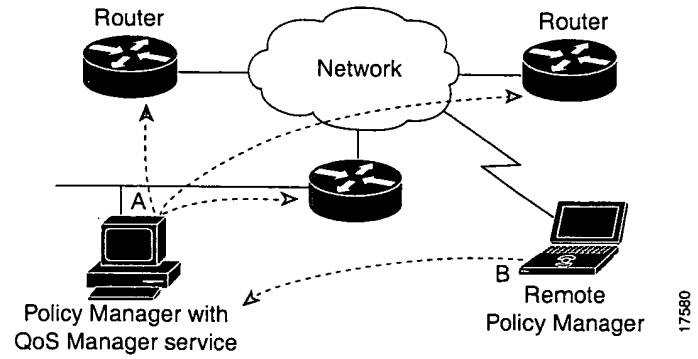
How Does QoS Policy Manager Deploy My QoS Policies?

QoS Policy Manager translates your policies into IOS software commands, and when you distribute your policies to the network, QoS Policy Manager creates the policies as ACLs on the devices. Through QoS Policy Manager, you can inspect the IOS software commands that will be used to configure the devices.

During policy distribution, you can view IOS software configuration messages as QoS Policy Manager configures each device, so that you can identify configuration successes and failures.

Figure 1-3 shows the relationship of QoS Policy Manager to the devices in the network. If you are using a remote version of QoS Policy Manager (B), it updates the network through the complete version (A). The QoS Manager service does the actual work of taking your policies, contacting the devices, and updating the device configurations.

Figure 1-3 QoS Policy Manager's Relationship to the Network



Does QoS Policy Manager Ensure That My Policies Are Consistent?

QoS Policy Manager does limited checking to ensure that your policies are consistent and can be implemented:

- You cannot define a policy or select a queuing technique that is not supported on the interface based on the selected IOS software version and device model.
- You cannot create policies that violate the rules of the queuing technique used on the interface. For example, QoS Policy Manager ensures that you do not create more than 16 queues on a custom queuing interface, and ensures that the queues do not allocate more than 95% of the interface's bandwidth.

QoS Policy Manager does not check to ensure your policies are logical. For example, if you have multiple policies on an interface, and the policies use the same filter conditions (thus selecting identical traffic), the second policy will never be applied (unless the first policy specifies that the interface should consider subsequent policies, which is a feature only available on certain IOS software version and device model combinations).

Thus, QoS Policy Manager ensures that your policies can be implemented, but does not ensure that your policies will have the effect you desire. The policies can not be implemented unless they meet a basic consistency.

How Does QoS Policy Manager Support My Existing ACL Policies?

If you have already defined access control lists (ACLs) on your devices, QoS Policy Manager does not translate them into the QoS Policy Manager database. If you want to manage those policies through QoS Policy Manager, you must recreate them in QoS Policy Manager.

If you leave the existing ACLs on the device, QoS Policy Manager does not change them or delete them. They remain defined on the device until you change or remove them using IOS software commands.

Planning for QoS Deployment

These topics help you decide how and where to deploy QoS in your network:

- Which Applications Benefit from QoS
- Which Interfaces Benefit from QoS
- Where to Deploy QoS in the Network

Which Applications Benefit from QoS

Some applications can benefit more from QoS techniques than other applications. The benefits you might get from QoS are dependent not only on the applications you use, but on the networking hardware and bandwidth available to you.

In general, QoS can help you solve two problems: constricted bandwidth and time sensitivity.

If you have insufficient bandwidth, either due to the lines you are leasing or the devices you have installed, QoS can help you allocate guaranteed bandwidth to your critical applications. Alternatively, you can limit the bandwidth for non-critical applications (such as FTP file transfers), so that your other applications have a greater amount of bandwidth available to them.

Some applications, such as video, require a certain amount of bandwidth for them to work in a usable manner. With QoS policies, you can guarantee the bandwidth required for these applications.

For time-sensitive applications, which are sensitive to timeouts or other delays, you can help the applications by coloring their traffic with higher priorities than your regular traffic. You can also define minimum bandwidth to help ensure the applications can deliver data in a timely fashion.

As you deploy QoS, identify the applications used on your network that are bandwidth or time sensitive, and also identify the applications that take more than their fair share of bandwidth. With this information, you can develop effective policies to improve the overall functioning of your network.

Which Interfaces Benefit from QoS

Any interface that is congested can benefit from QoS policies. LAN-WAN links are typical points of congestion, as data is moved between lines that have differing carrying capacity. These links might be the best place to start deploying QoS policies. However, the congestion points for your network might be anywhere. You evaluate interface points where packets most likely get dropped during peak traffic periods.

Where to Deploy QoS in the Network

Deploy QoS everywhere you have bandwidth contention. Traffic analysis of different times of the day is helpful in identifying potential bandwidth contention. Note the high traffic periods for peak and length. Also, identify critical data routing where packets may not be dropped.

More Information About Quality of Service

For more information about Quality of Service, see these papers on the web:

- Cisco IOS Technologies Quality of Service Overview: http://www.cisco.com/warp/public/732/Tech/quality.shtml

- Cisco IOS Quality of Service (requires Cisco Connection Online login): http://www.cisco.com/warp/customer/732/qos/index.html
- Bandwidth Management and Queuing: http://www.cisco.com/warp/public/731/Protocol/dlsw5_rg.htm
- Cisco—IOS Technologies—Quality—Random Early Detection: http://www.cisco.com/warp/public/732/Tech/red/
- Distributed WRED: http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.htm

Note For pages that require a Cisco Connection Online (CCO) login, you can register at the CCO web site at http://www.cisco.com/register/.

This paper has been submitted for presentation at INET'98, 21-24 July 1998, Geneva, Switzerland.

Portions of text in this paper has been extracted from "Quality of Service: Delivering QoS on the Internet and in Corporate Networks," by Paul Ferguson and Geoff Huston, published by John Wiley & Sons, January 1998, ISBN 0-471-24358-2.

Quality of Service in the Internet: Fact, Fiction, or Compromise?

Paul Ferguson

Cisco Systems, Inc.

400 Herndon Parkway

Herndon, Virginia 20170 USA

ferguson@cisco.com

Geoff Huston

Telstra Internet

5/490 Northbourne Ave.,

Dickson, ACT 2602, Australia

gih@telstra.net

Abstract *- The Internet has historically offered a single level of service, that of "best effort," where all data packets are treated with equity in the network. However, we are finding that the Internet itself does not offer a single level of service quality, and some areas of the network exhibit high levels of congestion and consequently poor quality, while other areas display consistent levels of high quality service. Customers are now voicing a requirement to define a consistent service quality they wish to be provided, and network service providers are seeking ways in which to implement such a requirement. This effort is happening within the umbrella called "Quality of Service" (QoS). Of course, this is now a phrase which has become overly used, often in vague, non-definitive references. QoS discussions currently embrace abstract concepts, varying ideologies, and moreover, lacks a unified definition on what QoS actually is and how it might be implemented. Subsequently, expectations regarding QoS have not been appropriately managed within the Internet community at-large on how QoS technologies might be realistically deployed on a global scale. A more important question is whether ubiquitous end-to-end QoS is even realistic in the Internet, given the fact that the decentralized nature of the Internet does not lend itself to homogenous mechanisms to differentiate traffic. This paper examines the various methods of delivering QoS in the Internet, and attempts to provide an objective overview on whether QoS in the Internet is fact, fiction, or a matter of compromise.*

Table of Contents

1 Introduction

It is hard to dismiss the entrepreneurial nature of the Internet today - this is no longer a research project. For most

organizations connected to the global Internet, it's a full-fledged business interest. Having said that, it is equally hard to dismiss the poor service quality that is frequently experienced - the rapid growth of the Internet, and increasing levels of traffic, make it difficult for Internet users to enjoy consistent and predictable end-to-end levels of service quality.

What causes poor service quality within the Internet? The glib, and rather uninformative response is "localized instances of substandard network engineering which is incapable of carrying high traffic loads."

Perhaps the more appropriate question is "what are the components of service quality and how can they be measured?" *Service quality* in the Internet can be expressed as the combination of network-imposed ***delay***, ***jitter***, ***bandwidth*** and ***reliability***.

Delay is the elapsed time for a packet to be passed from the sender, through the network, to the receiver. The higher the delay, the greater the stress that is placed on the transport protocol to operate efficiently. For the TCP protocol, higher levels of delay imply greater amounts of data held "in transit" in the network, which in turn places stress on the counters and timers associated with the protocol. It should also be noted that TCP is a "self-clocking" protocol, where the sender's transmission rate is dynamically adjusted to the flow of signal information coming back from the receiver, via the reverse direction acknowledgments (ACK's), which notify the sender of successful reception. The greater the delay between sender and receiver, the more insensitive the feedback loop becomes, and therefore the protocol becomes more insensitive to short term dynamic changes in network load. For interactive voice and video applications, the introduction of delay causes the system to appear unresponsive.

Jitter is the variation in end-to-end transit delay (in mathematical terms it is measurable as the absolute value of the first differential of the sequence of individual delay measurements). High levels of jitter cause the TCP protocol to make very conservative estimates of round trip time (*RTT*), causing the protocol to operate inefficiently when it reverts to timeouts to reestablish a data flow. High levels of jitter in UDP-based applications are unacceptable in situations where the application is real-time based, such as an audio or video signal. In such cases, jitter causes the signal to be distorted, which in turn can only be rectified by increasing the receiver's reassembly playback queue, which effects the delay of the signal, making interactive sessions very cumbersome to maintain.

Bandwidth is the maximal data transfer rate that can be sustained between two end points. It should be noted that this is limited not only by the physical infrastructure of the traffic path within the transit networks, which provides an upper bound to available bandwidth, but is also limited by the number of other flows which share common components of this selected end-to-end path.

Reliability is a commonly conceived as a property of the transmission system, and in this context, it can be thought of as the average error rate of the medium. Reliability can also be a byproduct of the switching system, in that a poorly configured or poorly performing switching system can alter the order of packets in transit, delivering packets to the receiver in a different order than that of the original transmission by the sender, or even dropping packets through transient routing loops. Unreliable or error-prone network transit paths can also cause retransmission of the lost packets. TCP cannot distinguish between loss due to packet corruption and loss due to congestion, and packet loss invokes the same congestion avoidance behavior response from the sender, causing the sender's transmit rates to be reduced by invoking congestion avoidance algorithms even though no congestion may have been experienced by the network. In the case of UDP-based voice and video applications, unreliability causes induced distortion in the original analog signal at the receiver's end.

Accordingly, when we refer to differentiated service quality, we are referring to differentiation of one or more of these four basic quality metrics for a particular category of traffic.

Given that we can define some basic parameters of service quality, the next issue is how is service quality implemented within the Internet?

The Internet is composed of a collection of routers and transmission links. Routers receive an incoming packet, determine the next hop interface, and place the packet on the output queue for the selected interface. Transmission links have characteristics of delay, bandwidth and reliability. Poor service quality is typically encountered when the level of traffic selecting a particular hop exceeds the transmission bandwidth of the hop for an extended period time. In such cases, the router's output queues associated with the saturated transmission hop begin to fill, causing additional transit delay (increased **jitter** and **delay**), until the point is reached where the queue is filled, and the router is then forced to discard packets (reduced **reliability**). This in turn forces adaptive flows to reduce their sending rate to minimize congestion loss, reducing the available **bandwidth** for the application. Poor service quality can be generated in other ways, as well. Instability in the routing protocols may cause the routers to rapidly alter their selection of the best next hop interface, causing traffic within an end-to-

end flow to take divergent paths, which in turn will induce significant levels of **jitter**, and an increased probability of out-of-order packet delivery (reduced **reliability**).

Accordingly, when we refer to the quality of a service, we are looking at these four metrics as the base parameters of quality, and it must be noted that there are variety of network events which can effect these parameter values.

Also, it should be noted that in attempting to take a uniform "best effort" network service environment and introduce structures which allow some form of service differentiation, the tools which allow such service environments to be constructed are configurations within the network's routers designed to implement one or more of the following -

- Signal the lower level transmission links to use a different transmission servicing criteria for particular service profiles,
- Alter the next hop selection algorithm to select a next hop which matches the desired service levels,
- Alter the router's queuing delay and packet discard algorithms such that packets are scheduled to receive transmission resources in accordance with their relative service level, and
- Alter the characteristics of the traffic flow as it enters the network to conform to a contracted profile and associated service level.

The "art" of implementing an effective QoS environment is to use these tools in a way which can construct robust differentiated service environments.

From this perspective, the concept of *service quality* is important to understand, as opposed to what most people call *quality of service*, or QoS. Service quality can be defined as delivering consistently predictable service, to include high network reliability, low delay, low jitter, and high availability. QoS, on other hand, can be interpreted as method to provide preferential treatment to some arbitrary amount of network traffic, as opposed to all traffic being treated as "best effort," and in providing such preferential treatment, attempting to increase the quality level of one of more of these basic metrics for this particular category of traffic. There are several tools available to provide this differentiation, ranging from preferential queuing disciplines to bandwidth reservation protocols, from ATM-layer congestion and bandwidth allocation mechanisms to traffic-shaping, each of which may be appropriate dependent on what problem is being solved. We do not see QoS as being principally concerned about attempting to deliver "guaranteed" levels of service to individual traffic flows within the Internet. While such network mechanisms may have a place within smaller network environments, the sheer size of today's Internet effectively precludes any QoS approach which attempts to reliably segment the network on a flow-by-flow basis. The major technology forces which have driven the explosive growth of the Internet as a communications medium are the use of stateless switching systems which provide variable best effort service levels to intelligent peripheral devices. Recent experience has indicated that this approach has extraordinary scaling properties, where the stateless switching architectures can scale easily into scales of gigabits per second, preserving a continued functionality where the unit cost of stateless switching has decreased at a level which is close to the basic scaling rate.

We also suggest that if a network cannot provide a reasonable level of service quality, then attempting to provide some method of differentiated QoS on the same infrastructure is virtually impossible. This is where traditional engineering, design, and network architectural principles play a significant role.

2 QoS and Network Engineering, Design, and Architecture

Before examining methods to introduce QoS into the Internet, it is necessary to examine methods of constructing a network that exemplifies sound network engineering principles - scalability, stability, availability, and predictability.

Typically, this exercise entails a conservative approach in designing and operating the network, undertaking measures to ensure that the routing system within the network remains stable, and ensuring that peak level traffic flows sit comfortably within the bandwidth and switching capabilities of the network. Unfortunately, some of these seminal principles are ignored in favor of maximizing revenue potential. For example, it is not uncommon for an ISP (Internet Service Provider) to oversubscribe an access aggregation point by a factor of 25 to 1, especially in the case of calculating the number of subscribers compared to the number of available modem ports, nor is it uncommon to see interprovider exchange points experiencing peak packet drop rates in excess of 20% of all transit traffic. In a single quality best effort environment, the practice of oversubscription allows the introduction of additional load into the network at the expense of marginal degradation to all existing active subscribers. This can be a very dangerous practice, and if miscalculated, can result in seriously degraded service performance (due to induced congestion) for all subscribers. Therefore, oversubscription should

not be done arbitrarily.

Network engineering is arguably a compromise between engineering capabilities for the average load levels, and engineering capabilities which are intended to handle peak load conditions. In the case of dimensioning access ports, close attention must be paid to user traffic characteristics and modem pool port usage, based on time-of-day and day-of-week, for an extended period prior to making such an engineering commitment. Even after such traffic analysis has been undertaken, the deployed configuration should be closely monitored on a continuing and consistent basis to detect changes in usage and characteristics. It is very easy to assume that only a fraction of subscribers may be active at any given time, or to assume average and peak usage rates, but without close observation and prior traffic sampling, a haphazard assumption could dramatically affect the survival of your business.

Equally, it is necessary for the network operator to understand the nature, size, and timing of traffic flows that are carried across the network's transmission systems. While a critical component of traffic analysis is the monitoring of the capacity on individual transmission links, monitoring the dispersion of end-to-end traffic flows allows the network operator to ensure that the designed transmission topology provides an efficient carriage for the data traffic. It also attempts to avoid the situation where major traffic flows are routed sub-optimally across multiple hops, incurring additional cost and potentially imposing a performance penalty through the transit through additional routing points.

The same can be said of maintaining stability in the ISP's network routing system. Failure to create a highly stable routing system can result in destinations being intermittently unreachable, and ultimately frustrating customers. Care should be taken on all similar infrastructure and "critical" service issues, such as DNS (Domain Name System) services. The expertise of the engineering and support staff will be reflected in the service quality of the network, like it or not.

Having said that, it is not difficult to understand that poorly designed networks do not lend themselves to QoS scenarios, due to the fact that if acceptable levels of service quality cannot be maintained, then it is quite likely that adding QoS in an effort to create some level of service differentiation will never be effective. Granted, it may allow the network performance to degrade more "gracefully" in times of severe congestion for some applications operated within the group of elevated QoS customers, but limiting the impact of degradation for some, at the cost of increasing the impact for the remainder of the customer base, is not the most ingenious or sensible use for QoS mechanisms. The introduction of QoS differentiation into the network is only partially effective if those customers who do not subscribe to a QoS service are adversely impacted those who do choose to subscribe to such a service. After all, if non-QoS subscribers are negatively impacted, they will seek other service providers for their connectivity, or they will be forced to subscribe to the QoS service to obtain an acceptable service level. This last sentence requires a bit of unconventional logic, since not all subscribers can realistically be QoS subscribers - this violates some of the most fundamental QoS strategies.

The design principles which are necessary to support effective QoS mechanisms can be expressed in terms of the four base service quality parameters noted in the previous section - delay, jitter, bandwidth, and reliability. In order to minimize delay, the network must be based upon a transmission topology which reflects the pattern of end-to-end traffic flows, and a routing system design which attempts to localize traffic such that minimal distance paths are always preferred. In order to minimize jitter, the routing system must be held in as stable a state as possible. Router queue depths must also be configured such that they remain within the same order of magnitude in size as the delay bandwidth product of the transmission link that is fed by the queue. Also, unconditional preferential queuing mechanisms should be avoided in favor of weighting or similar fair access queue mechanisms, to ensure that all classes of traffic are not delayed indefinitely while awaiting access to the transmission resources. Selection of maximum transfer unit (MTU) sizes should also be undertaken to avoid MTU's which are very much greater than the delay bandwidth product of the link - again as a means of minimizing levels of network-induced jitter.

In terms of overall reliability, the onus is on the network architect to use transmission media which have a very low intrinsic bit error rates, and to use router components which have high levels of availability and stability. Care should also be taken to ensure that the routing system is configured to provide deterministic outcomes, minimizing the risk of packet reordering. Transmission capacity, or bandwidth, should be engineered to minimize the level of congestion-induced packet loss within the routers. This is perhaps not so straightforward as it sounds, given that transmission capacity is one of the major cost elements for an Internet network service provider, and the network architect typically has to assess the trade-off between the cost performance of the network, and the duration and impact of peak load conditions on the network. Typically, the network architect looks for average line utilization, and "busy hour" to "average hour" utilization ratios to provide acceptable levels of economic performance, while looking at busy hour performance figures to ensure that the network does not revert into a condition of congestion collapse at the points in time where usage is at a maximum.

It is only after these basic design steps have been undertaken, and a basic level of service quality achieved within the

network, that the issue of QoS (or service level differentiation) can be examined in any productive manner. The general conclusion here is that you cannot introduce QoS mechanisms to salvage a network which is delivering very poor levels of service. In order to be effective, QoS mechanisms need to be implemented in a network that is soundly engineered, and which operates in a stable fashion under all levels of offered load.

3 QoS Tools

Now that we have provided a framework definition for QoS, there are several mechanisms (and architectural implementations) which can provide differentiation for traffic in the network. We break these mechanisms into three basic groups, which align with the lower three layers of the OSI reference model - the Physical, Link, and Network layers [Figure 1].

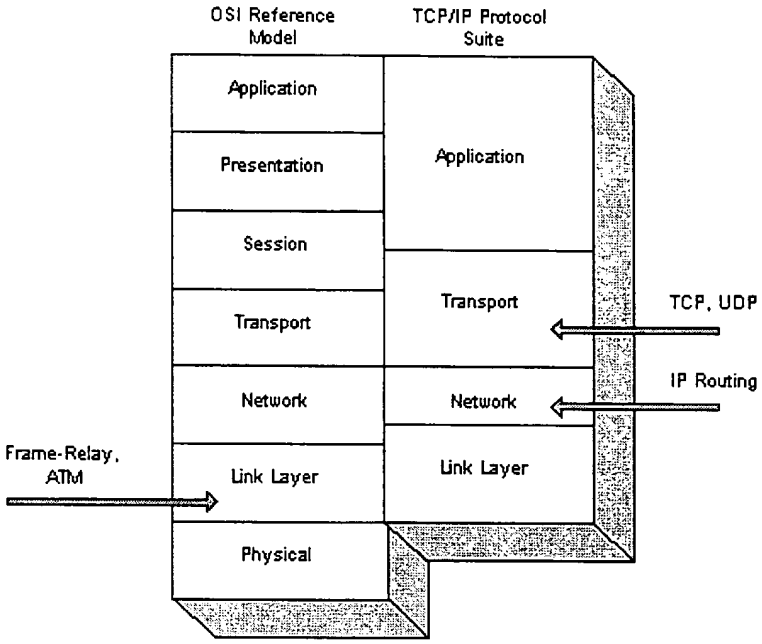


Figure 1

3.1 Physical Layer Mechanics

The physical layer (also referred to as L1, or Layer 1), consists of the physical wiring, fiber optics, and the transmission media in the network itself. It is reasonable to ask how Layer 1 physical media figures within the QoS framework, but the time-honored practice of constructing diverse physical paths in a network is, perhaps ironically, a primitive method of providing differentiated service levels. In some cases, diverse paths are constructed primarily to be used by network layer routing to provide for redundant availability, should the primary physical path fail for some reason, although often the temptation to share the load across the primary and backup paths is overwhelming. This can lead to adverse performance where, for example, having more than one physical path to a destination can theoretically allow for some arbitrary amount of network traffic to take the primary low-delay, high-bandwidth path, while the balance of the traffic takes a backup path which may have different delay and bandwidth properties. In turn, such a configuration leads to reduced reliability and increased jitter within the network as a consequence, unless the routing profile has been carefully constructed to stabilize the traffic segmentation between the two paths.

3.1.1 Alternate Physical Paths

While the implementation of provisioning diverse physical paths in a network is usually done to provide for backup and redundancy, this can also be used to provide differentiated services if the available paths each have differing characteristics. In [Figure 2], for example, best-effort traffic could be forwarded by the network layer devices (routers) along the lower-speed path, while higher priority (QoS) traffic could be forwarded along the higher-speed path.

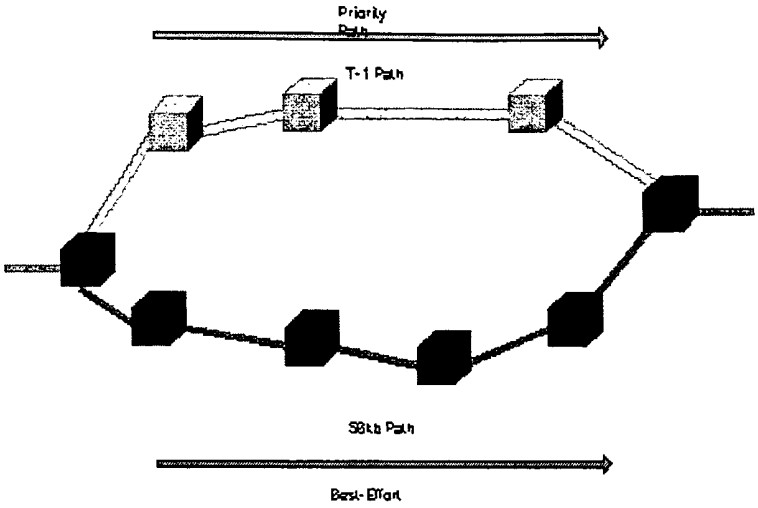


Figure 2

Alternatively, the scenario could be a satellite path complemented by a faster terrestrial cable path. Best effort traffic would be passed along the higher delay satellite path, while priority traffic would be routed along the terrestrial cable system.

Certainly, this type of approach is indeed primitive, and not without its pitfalls.

Destination-Based Routing and Path Selection

The method in which IP packets are forwarded in the Internet is based on the destination contained in the packet header. This is termed *destination-based routing*, and the byproduct of this mode of packet forwarding is that since packets are generally switched based on a local decision of the best path to the IP destination address contained in the IP packet header, the mechanisms which do exist to forward traffic based on its IP source address are not very robust. Accordingly, it is difficult to perform outbound path selection based on the characteristics of the traffic source. The default form of path selection is based on the identity of the receiver. This implies that a QoS differentiation mechanism using path selection would be most efficiently implemented on selection of incoming traffic, while outgoing traffic would adopt the QoS parameters of the receiver. A destination-based routed network cannot control the QoS paths of both incoming and outgoing traffic to any particular location. Each QoS path will be determined as a destination-based path selection, leading to the observation that in a heterogeneous QoS environment, asymmetric quality parameters on incoming and outgoing data flows will be observed. This is a significant issue for unidirectional UDP-based traffic flows, where it becomes the receiver who controls the quality level of the transmission, not the sender. It is also significant for TCP, where the data flow and the reverse ACK flows may take paths of differing quality. Given that the sender adapts its transmission rate via signaling ,which transits the complete round trip, the resultant quality of the entire flow is influenced by the lower of the two quality levels of the forward and reverse paths.

TCP and Symmetric Path Selection

Reliable traffic transmission (TCP) requires a bidirectional data flow - sessions which are initiated and established by a particular host (sender) generally require control traffic (e.g. explicit acknowledgments, or the notification that acknowledgments were not processed by the receiver) to be returned from the destination (receiver). This reverse data flow is used to determine the transmission success, out-of-order traffic reception at the receiver, transmission rate adaptation, or other maintenance and control signals, in order to operate correctly. In essence, this reverse flow allows the sender to infer what is happening along the forward path and at the receiver, allowing the sender to optimize the flow data rate to fully utilize its fair share of the forward paths resource level. Therefore, for a reliable traffic flow which is transmitted along a particular path at a particular differentiated quality level, the flow will need to have its return traffic flow traverse the same path at the same quality level if optimal flow rates are to be reliably maintained (this routing characteristic is also known as symmetric paths). Asymmetric paths in the Internet continues to be a troubling issue with regards to traffic which is sensitive to induced delay and differing service quality levels, which effectively distort the signal being generated by the receiver. This problem is predominantly due to local routing policies in individual administrative domains through which traffic in the Internet must traverse. It is especially unrealistic to expect path symmetry in the Internet, at least for the foreseeable future.

The conclusion is that path diversity does allow for differentiated service levels to be constructed from the different delay, bandwidth, and load characteristics of the various paths. However, for reliable transmission applications, this differentiation is relatively crude.

3.2 Link Layer Mechanics

There exists a belief that traffic service differentiation can be provided with specific link layer mechanisms (also referred to as Layer 2, or L2), and traditionally this belief in differentiation has been associated with Asynchronous Transfer Mode (ATM) and Frame Relay in the wide area network (WAN), and predominantly with ATM in the local area network (LAN) campus. A brief overview is provided here of how each of these technologies provide service differentiation, and additionally, an we provide an overview of the newer IEEE 802.1p mechanics which may be useful to provide traffic differentiation on IEEE 802 style LAN media.

3.2.1 ATM

ATM is one of the few transmission technologies which provides data-transport speeds in excess of 155 Mbps today. As well as a high-speed bit-rate clock, ATM also provides a complex subset of traffic-management mechanisms, Virtual Circuit (VC) establishment controls, and various associated QoS parameters for these VC's. It is important to understand why these underlying transmission QoS mechanisms are not being exploited by a vast number of organizations that are using ATM as a data-transport tool for Internet networks in the wide area. The predominate use of ATM in today's Internet networks is simply because of the high data-clocking rate and multiplexing flexibility available with ATM implementations. There are few other transmission technologies which provide a such high speed bit-rate clock.

However, it is useful to examine the ATM VC service characteristics and examine their potential applicability to the Internet environment.

3.2.1.1 Constant Bit Rate (CBR)

The ATM CBR service category is used for virtual circuits that transport traffic at a consistent bit rate, where there is an inherent reliance on time synchronization between the traffic source and destination. CBR is tailored for any type of data for which the end-systems require predictable response time and a static amount of bandwidth continuously available for the lifetime of the connection. The amount of bandwidth is characterized by a Peak Cell Rate (PCR). These applications include services such as video conferencing, telephony (voice services), or any type of on-demand service, such as interactive voice and audio. For telephony and native voice applications, AAL1 (ATM Adaptation Layer 1) and CBR service is best suited to provide low-latency traffic with predictable delivery characteristics. In the same vein, the CBR service category typically is used for circuit emulation. For multimedia applications, such as video, you might want to choose the CBR service category for a compressed, frame-based, streaming video format over AAL5 for the same reasons.

3.2.1.2 Real-Time and Non-Real-Time Variable Bit Rate (rt- and nrt-VBR)

The VBR service categories are generally used for any class of applications that might benefit from sending data at variable rates to most efficiently use network resources. Real-Time VBR (rt-VBR), for example, is used for multimedia applications with *lossy* properties, applications that can tolerate a small amount of cell loss without noticeably degrading the quality of the presentation. Some multimedia protocol formats may use a lossy compression scheme that provides these properties. Non-Real-Time VBR (nrt-VBR), on the other hand, is predominantly used for transaction-oriented applications, where traffic is sporadic and bursty.

The rt-VBR service category is used for connections that transport traffic at variable rates - traffic that relies on accurate timing between the traffic source and destination. An example of traffic that requires this type of service category are variable rate, compressed video streams. Sources that use rt-VBR connections are expected to transmit at a rate that varies with time (e.g., traffic that can be considered bursty). Real-time VBR connections can be characterized by a Peak Cell Rate (PCR), Sustained Cell Rate (SCR), and Maximum Burst Size (MBS). Cells delayed beyond the value specified by the maximum CTD (Cell Transfer Delay) are assumed to be of significantly reduced value to the application, thus, delay is indeed considered in the rt-VBR service category.

The nrt-VBR service category is used for connections that transport variable bit rate traffic for which there is no inherent reliance on time synchronization between the traffic source and destination, but there is a need for an attempt at a guaranteed

bandwidth or latency. An application that might require an nrt-VBR service category is Frame Relay interworking, where the Frame Relay CIR (Committed Information Rate) is mapped to a bandwidth guarantee in the ATM network. No delay bounds are associated with nrt-VBR service.

3.2.1.3 Available Bit Rate (ABR)

The ABR service category is similar to nrt-VBR, because it also is used for connections that transport variable bit rate traffic for which there is no reliance on time synchronization between the traffic source and destination, and for which no required guarantees of bandwidth or latency exist. ABR provides a best-effort transport service, in which flow-control mechanisms are used to adjust the amount of bandwidth available to the traffic originator. The ABR service category is designed primarily for any type of traffic that is not time sensitive and expects no guarantees of service. ABR service generally is considered preferable for TCP/IP traffic, as well as other LAN-based protocols, that can modify its transmission behavior in response to the ABR's rate-control mechanics.

ABR uses Resource Management (RM) cells to provide feedback that controls the traffic source in response to fluctuations in available resources within the interior ATM network. The specification for ABR flow control uses these RM cells to control the flow of cell traffic on ABR connections. The ABR service expects the end-system to adapt its traffic rate in accordance with the feedback so that it may obtain its fair share of available network resources. The goal of ABR service is to provide fast access to available network resources at up to the specified Peak Cell Rate (PCR).

3.2.1.4 Unspecified Bit Rate (UBR)

The UBR service category also is similar to nrt-VBR, because it is used for connections that transport variable bit rate traffic for which there is no reliance on time synchronization between the traffic source and destination. However, unlike ABR, there are no flow-control mechanisms to dynamically adjust the amount of bandwidth available to the user. UBR generally is used for applications that are very tolerant of delay and cell loss. UBR has enjoyed success in Internet LAN and WAN environments for store-and-forward traffic, such as file transfers and e-mail. Similar to the way in which upper-layer protocols react to ABR's traffic-control mechanisms, TCP/IP and other LAN-based traffic protocols can modify their transmission behavior in response to latency or cell loss in the ATM network.

3.2.1.5 The Misconceptions about ATM QoS

Several observations must be made to realize the value of ATM QoS and its associated complexity. This section attempts to provide an objective overview of the problems associated with relying solely on ATM to provide QoS in the network. However, it sometimes is difficult to quantify the significance of some issues because of the complexity involved in the ATM QoS delivery mechanisms, and their interactions with higher-layer protocols and applications. In fact, the inherent complexity of ATM and its associated QoS mechanisms may be a big reason why many network operators are reluctant to implement those QoS mechanisms.

While the underlying recovery mechanism of ATM cell loss is signaling, the QoS structures for ATM VC's are excessively complex, and that when tested against the principle of Occam's Razor (a popular translation frequently used in the engineering community for years is, "All things being equal, choose the solution that is simpler"), ATM by itself would not be the choice for QoS services, simply because of the complexity involved, compared to other technologies that provide similar results. Having said that, however, the application of Occam's Razor does not provide assurances that the desired result will be delivered - instead, it simply expresses a preference for simplicity.

ATM enthusiasts correctly point out that ATM is complex for good reason - in order to provide predictive, proactive, and real-time services, such as dynamic network resource allocation, resource guarantees, virtual circuit rerouting, and virtual circuit path establishment to accommodate subscriber QoS requests, ATM's complexity is unavoidable. The underlying model of ATM is a heterogeneous client population where the real time service models assume simple clients which are highly intolerant of jitter, while the adaptive models assume very highly sophisticated clients which can opportunistically tune their data rates to variations in available capacity which may fluctuate greatly within time frames well inside normal end-to-end round trip times.

It also has been observed that higher-layer protocols, such as TCP/IP, provide the end-to-end transportation service in most cases, so that although it is possible to create QoS services in a lower layer of the protocol stack (namely ATM in this case), such services may cover only part of the end-to-end data path. This gets to the heart of the problem in delivering QoS with ATM, when the true end-to-end bearer service is not pervasive ATM. Such partial QoS measures often have their effects

masked by the effects of the traffic distortion created from the remainder of the end-to-end path in which they do not reside, and hence the overall outcome of a partial QoS structure often is ineffectual. In other words, if ATM is not pervasively deployed end-to-end in the data path, efforts to deliver QoS using ATM can be unproductive. There is traffic distortion introduced into the ATM landscape by traffic-forwarding devices which service the ATM network and upper-layer protocols such as IP, TCP, and UDP, as well as other upper-layer network protocols. Queuing and buffering introduced into the network by routers and non-ATM-attached hosts skew the accuracy with which the lower-layer ATM services calculate delay and delay variation (jitter).

On a related note, some have suggested that most traffic on ATM networks would be primarily UBR or ABR connections, because higher-layer protocols and applications cannot request specific ATM QoS service classes, and therefore cannot fully exploit the QoS capabilities of the VBR service categories. A cursory examination of deployed ATM networks and their associated traffic profiles reveals that this is indeed the case, except in the rare instance when an academic or research organization has developed its own native "ATM-aware" applications that can fully exploit the QoS parameters available to the rt-VBR and nrt-VBR service categories. Although this certainly is possible, and has been done on several occasions, real-world experience reveals that this is the proverbial exception and not the rule.

It is interesting to note the observations published by Jagannath and Yin [1], which suggests that "it is not sufficient to have a lossless ATM subnetwork from the end-to-end performance point of view," especially in the case of ABR services. This observation is due to the fact that two distinct control loops exist - ABR and TCP [Figure 3]. Although it generally is agreed that ABR can effectively control the congestion in the ATM network, ABR flow control simply pushes the congestion to the edges of the network (i.e., the routers), where performance degradation or packet loss may occur as a result. Jagannath and Yin also point out that "one may argue that the reduction in buffer requirements in the ATM switch by using ABR flow control may be at the expense of an increase in buffer requirements at the edge device (e.g., ATM router interface, legacy LAN to ATM switches)." Because most applications use the flow control provided by TCP, one might question the benefit of using ABR flow control at the subnetwork layer, because UBR (albeit with Early Packet Discard) is equally effective and much less complex. ABR flow control also may result in longer feedback delay for TCP control mechanisms, and this ultimately exacerbates the overall congestion problem in the network.

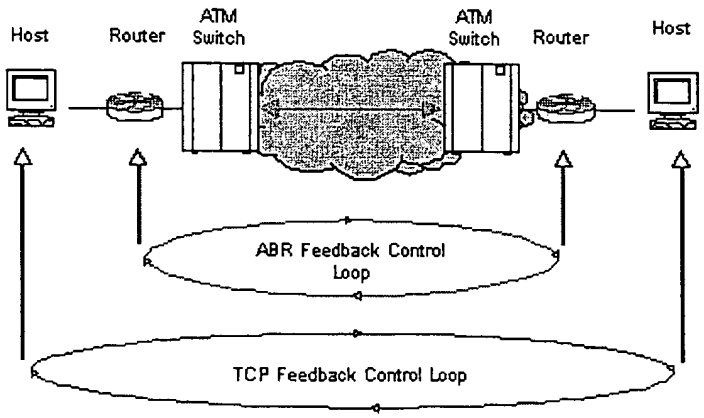


Figure 3

Aside from traditional data services that may use UBR, ABR, or VBR services, it is clear that circuit-emulation services which may be provisioned using the CBR service category can certainly provide the QoS necessary for telephony communications. However, this becomes an exercise in comparing apples and oranges. Delivering voice services on virtual digital circuits using circuit emulation is quite different than delivering packet-based data found in local-area and wide-area networks. Providing QoS in these two environments is substantially different - it is substantially more difficult to deliver QoS for data, because the higher-layer applications and protocols do not provide the necessary hooks to exploit the QoS mechanisms in the ATM network. As a result, an intervening router must make the QoS request on behalf of the application, and thus the ATM network really has no way of discerning what type of QoS the application may truly require. This particular deficiency has been the topic of recent research and development efforts to address this shortcoming and investigate methods of allowing the end-systems to request network resources using RSVP, and then map these requests to native ATM QoS service classes as appropriate.

The QoS objective for networks similar in nature to the Internet lies principally in directing the network to alter the switching behavior at the IP layer, so that certain IP packets are delayed or discarded at the onset of congestion or delay, or completely avoid if at all possible, the impact of congestion on other classes of IP traffic. When looking at IP-over-ATM, the issue (as with IP-over-Frame Relay) is that there is no mechanism for mapping such IP-level directives to the ATM level, nor is it desirable, given the small size of ATM cells and the consequent requirement for rapid processing or discard. Attempting to increase the complexity of the ATM cell discard mechanics to the extent necessary to preserve the original IP QoS directives by mapping them into the ATM cell is arguably counterproductive.

Thus, it appears that the default IP QoS approach is best suited to IP-over-ATM. It also stands to reason that if the ATM network is adequately dimensioned to handle burst loads without the requirement to undertake large-scale congestion avoidance at the ATM layer, there is no need for the IP layer to invoke congestion-management mechanisms. Thus, the discussion comes full circle to an issue of capacity engineering, and not necessarily one of QoS within ATM.

3.2.2 Frame Relay

Frame Relay's origins lie in the development of ISDN (Integrated Services Digital Network) technology, where Frame Relay originally was seen as a packet-service technology for ISDN networks. The Frame Relay rationale proposed was the perceived need for the efficient relaying of HDLC framed data across ISDN networks. With the removal of data link-layer error detection, retransmission, and flow control, Frame Relay opted for end-to-end signaling at the transport layer of the protocol stack to undertake these functions. This allows the network switches to consider data-link frames as being forwarded without waiting for positive acknowledgment from the next switch. This in turn allows the switches to operate with less memory and to drive faster circuits with the reduced switching functionality required by Frame Relay.

3.2.2.1 Frame Relay Rate Management Control Structures

Frame Relay is a link layer protocol which attempts to provide a simple mechanism for arbitration of network oversubscription. Frame Relay decouples the characteristics of the network access link from the characteristics of the virtual circuits which connect the access system to its group peers. Each virtual circuit is configured with a traffic committed information rate (CIR), which conforms to a commitment on the part of the network to provide traffic delivery. However, any virtual circuit can also accept overflow traffic levels - bursts which may transmit up to the rate of the access link. Such excess traffic is marked by the network access gateway using a single bit indicated in the Frame Relay frame header, termed the "Discard Eligible" (DE) bit.

The interior of the network uses three basic levels of threshold to manage switch queue congestion. At the first level of queue threshold, the network starts to mark frames with Explicit Congestion Notification (ECN) bits. Frame relay congestion control is handled in two ways - congestion avoidance and congestion recovery. Congestion avoidance consists of a Backward Explicit Congestion Notification (BECN) bit and a Forward Explicit Congestion Notification (FECN) bit, both which are also contained in the Frame Relay frame header. The BECN bit provides a mechanism for any switch in the frame relay network to notify the originating node (sender) of potential congestion when there is a build-up of queued traffic in the switch's queues. This informs the sender that the transmission of additional traffic (frames) should be restricted. The FECN bit notifies the receiving node of potential future delays, informing the receiver to use possible mechanisms available in a higher-layer protocol to alert the transmitting node to restrict the flow of frames. The implied semantics of the congestion notification signaling is to notify senders and receivers to reduce their transmission rates to the CIR levels, although this action is not forced upon them. At the second level of queue threshold, the switch discards packets which are marked as DE, honoring its commitment to traffic which conforms to committed information rates on each circuit.

The basic premise within Frame Relay networks is that the switching fabric is dimensioned at such a level that it can fulfill its obligations of committed traffic flows. If this is not the case, and discarding of all discard eligible traffic fails to remove the condition which is causing switch congestion, the switch passes the third threshold of the queue, where it discards frames which form part of committed flow rates.

Frame Relay allows a basic level of oversubscription of basic point-to-point virtual circuits, where individual flows can increase their transfer rate, with the intent of occupying otherwise idle transmission capacity which is not being utilized by other virtual circuits which share the same transmission paths. When the sender is not using all of the committed rate within any of the configured virtual circuits, other VC's can utilize the transmission space with discard eligible frames.

Frame Relay indicates that it is possible to provide reasonable structures of basic service commitment, together with the added capability of provision of overcommitment using a very sparse link level signaling set - the DE, FECN, and BECN

bits.

3.2.2.2 Frame Relay and Internet QoS

Frame Relay is certainly a good example of what is possible with relatively sparse signaling capability. However, the match between Frame Relay as a link layer protocol, and QoS mechanisms for the Internet, is not a particularly good one.

Frame Relay networks operate within a locally defined context of using selective frame discard as a means of enforcing rate limits on traffic as it enters the network. This is done as the primary response to congestion. The basis of this selection is undertaken without respect to any hints provided by the higher-layer protocols. The end-to-end TCP protocol uses packet loss as the primary signaling mechanism to indicate network congestion, but it is recognized only by the TCP session originator. The result is that when the network starts to reach a congestion state, the method in which end-system applications are degraded matches no particular imposed policy, and in this current environment, Frame Relay offers no great advantage over any other link layer technology in addressing this.

One can make the observation that in a heterogeneous network that uses a number of link layer technologies to support end-to-end data paths, the Frame Relay ECN and DE bits are not a panacea - they do not provide for end-to-end signaling, and the router is not the system that manages either end of the end-to-end protocol stack. The router is more commonly performing IP packet into Frame Relay encapsulation. With this in mind, a more functional approach to user-selection of DE traffic is possible, one that uses a field in the IP header to indicate a defined quality level via a single discard eligibility field, and allow this designation to be carried end-to-end across the entire network path. With this facility, it then is logical to allow the ingress IP router (which performs the encapsulation of an IP datagram into a Frame Relay frame) to set the DE bit according to the bit setting indicated in the IP header field, and then pass the frame to the first-hop Frame Relay switch, which then can confirm or clear the DE bit in accordance with locally configured policy associated with the per-VC CIR.

The seminal observation regarding the interaction of QoS mechanisms within various levels of the model of the protocol stack is that without coherence between the link layer transport signaling structures and the higher-level protocol stack, the result, in terms of consistency of service quality, is completely chaotic.

3.2.3 IEEE 802.1p

It should be noted that an interesting set of proposed enhancements is being reviewed by the IEEE 802.1 Internetworking Task Group. These enhancements would provide a method to identify 802-style frames based on a simple priority. A supplement to the original IEEE MAC Bridges standard [2], the proposed 802.1p specification [3] provides a method to allow preferential queuing and access to media resources by traffic class, on the basis of a "priority" value signaled in the frame. The IEEE 802.1p specification, if adopted, will provide a way to transport this value (called *user_priority*) across the subnetwork in a consistent method for Ethernet, token ring, or other MAC-layer media types using an extended frame format. Of course, this also implies that 802.1p-compliant hardware may have to be deployed to fully realize these capabilities.

The current 802.1p draft defines the *user_priority* field as a 3-bit value, resulting in a variable range of values between 0 and 7 decimal, with 7 indicating the highest relative priority and 0 indicating the lowest relative priority. The IEEE 802.1p proposal does not make any suggestions on how the *user_priority* should be used by the end-system or by network elements - it only suggests that packets may be queued by LAN devices based on their relative *user_priority* values.

While it is clear that the 802.1p *user_priority* may indeed prove to be useful in some QoS implementations, it remains to be seen how it will be most practically beneficial. At least one proposal exists [4] which suggests how the 802.1p *user_priority* values may be used in conjunction with the Subnet Bandwidth Manager (SBM), a proposal that allows LAN switches to participate in RSVP signaling and resource reservation objectives [5]. However, bearing in mind that the widespread deployment of RSVP in the global Internet is not wholly practical (as discussed in more detail below), it remains to be seen how QoS implementations will use this technology.

3.3 Network and Transport Layer Mechanics

In the global Internet, it is undeniable that the common bearer service is the TCP/IP protocol suite, therefore, IP is indeed the common denominator. (The TCP/IP protocol suite is commonly referred to simply as *IP* - this has become the networking

vernacular used to describe IP, as well as ICMP, TCP, and UDP.) This thought process has several supporting lines of reason. The common denominator is chosen in the hope of using the most pervasive and ubiquitous protocol in the network, whether it be Layer 2 or Layer 3 (the network layer). Using the most pervasive protocol makes implementation, management, and troubleshooting much easier and yields a greater possibility of successfully providing a QoS implementation that actually works.

It is also the case that this particular technology operates in an end-to-end fashion, using a signaling mechanism that spans the entire traversal of the network in a consistent fashion. IP is the end-to-end transportation service in most cases, so that although it is possible to create QoS services in substrate layers of the protocol stack, such services only cover part of the end-to-end data path. Such partial measures often have their effects masked by the effects of the signal distortion created from the remainder of the end-to-end path in which they are not present, or introduce other signal distortion effects, and as mentioned previously, the overall outcome of a partial QoS structure is generally ineffectual.

When the end-to-end path does not consist of a single pervasive data-link layer, any effort to provide differentiation within a particular link-layer technology most likely will not provide the desired result. This is the case for several reasons. In the Internet, for example, an IP packet may traverse any number of heterogeneous link-layer paths, each of which may (or may not) possess characteristics that inherently provide methods to provide traffic differentiation. However, the packet also inevitably traverses links that cannot provide any type of differentiated services at the link layer, rendering an effort to provide QoS solely at the link layer an inadequate solution.

It should also be noted that the Internet today carries three basic categories of traffic, and any QoS environment must recognize and adjust itself to these three basic categories. The first category is **long held adaptive reliable traffic flows**, where the end-to-end flow rate is altered by the end points in response to network behavior, and where the flow rate attempts to optimize itself in an effort to obtain a fair share of the available resources on the end-to-end path. Typically, this category of traffic performs optimally for long held TCP traffic flows. The second category of traffic is a boundary case of the first category, **short duration reliable transactions**, where the flows are of very short duration, and the rate adaptation does not get established within the lifetime of the flow, so that the flow sits completely within the startup phase of the TCP adaptive flow control protocol. The third category of traffic is an **externally controlled load unidirectional traffic flow**, which is typically a result of compression of a real time audio or video signal, where the peak flow rate may equal the basic source signal rate, and the average flow rate is a byproduct of the level of signal compression used, and the transportation mechanism is an unreliable traffic flow with a UDP unicast flow model. Within most Internet networks today, empirical evidence indicates that the first category of traffic accounts for less than 1% of all packets, but as the data packets are typically large, this application accounts for some 20% of the volume of data. The second category of traffic is most commonly generated by World Wide Web servers using the HTTP/1.0 application protocol, and this traffic accounts for some 60% of all packets, and a comparable relative level of volume of data carried. The third category accounts for some 10% of all packets, and as the average packet size is less than 1/3 of the first two flow types, it currently accounts for some 5% of the total data volume.

In order to provide elevated service quality to these three common traffic flow types, there are three different engineering approaches that must be used. To ensure efficient carriage of long held, high volume TCP flows, requires the network to offer consistent signaling to the sender regarding the onset of congestion loss within the network. To ensure efficient carriage of short duration TCP traffic requires the network to avoid sending advance congestion signals to the flow end-points. Given that these flows are of short duration and low transfer rate, any such signaling will not achieve any appreciable load shedding, but will substantially increase the elapsed time that the flow is held active, which results in poor delivered service without any appreciable change in the relative allocation of network resources to service clients. To ensure efficient carriage of the externally clocked UDP traffic requires the network to be able to, at a minimum, segment the queue management of such traffic from adaptive TCP traffic flows, and possibly replace adaptation by advance notification and negotiation. Such a notification and negotiation model could allow the source to specify its traffic profile in advance, and have the network respond with either a commitment to carry such a load, or indicate that it does not have available resources to meet such an additional commitment.

As a consequence, it should be noted that no single transport or network layer mechanism will provide the capabilities for differentiated services for all flow types, and that a QoS network will deploy a number of mechanisms to meet the broad range of customer requirements in this area.

3.3.1 TCP Congestion Avoidance

As noted above, there are two major types of traffic flow in the Internet today. One is an adaptive-rate, reliable transmission, control-mediated traffic flow using TCP. The other is externally clocked, unreliable data flows which typically use UDP.

Here, we look at QoS mechanisms for TCP, but to preface this it is necessary to briefly describe the behavior of TCP itself in terms of its rate control mechanisms.

TCP uses a rate control mechanism to achieve a sustainable steady state network load. The intent of the rate control mechanism is to reach a state where the sender injects a new data packet into the network at the same time as the receiver removes a data packet. However, this is modified by a requirement to allow dynamic rate probing, so that the sender attempts to inject slightly more data than is being removed by the receiver, and when this rate probing results in congestion, the sender will reduce its rate and again probe upwards to find a stable operating point. This is accomplished in TCP using two basic mechanisms.

The first mechanism used by TCP is *slow start*, where the connection is initialized with the transmission of a single segment. Each time the sender receives an ACK from the receiver, the congestion window (*cwnd*) is increased by one segment size. This effectively doubles the transmission rate for each round trip time (*RTT*) cycle - the sender transmits a single packet and awaits the corresponding ACK. A TCP connection commences with a initial *cwnd* value of 1, and a single segment is sent into the network. The sender than awaits the reception of the matching ACK from the receiver. When received, the *cwnd* is opened from 1 to 2, allowing 2 packets to be sent. When each ACK is received from these two segments, the congestion window is incremented. The value of *cwnd* is then 4, allowing 4 packets to be sent, and so on (receivers using delayed ACK will moderate this behavior such that the rate increase will be slightly less than doubled for each *RTT*).

This behavior will continue until the transfer completes, or the sender's buffer space is exhausted, in which case the sender is transmitting at its maximal possible rate across the network's selected path, given the delay bandwidth product of the path, or an intermediate router in the path experiences queue exhaustion and packets are dropped. Given that the algorithm tends to cluster transmission events at epoch intervals of the *RTT*, such overload of the router queue structure is highly likely when the path delay is significant.

The second TCP rate control mechanism is termed *congestion avoidance*. In the event of packet loss, as signaled by the reception of duplicate ACK's, the value of *cwnd* is halved, and this value is saved as the threshold value to terminate the *slow start* algorithm (*ssthresh*). When *cwnd* exceeds this threshold value, the window is increased in a linear fashion, opening the window by one segment size in each *RTT* interval. The value of *cwnd* is bought back to 1 when the end-to-end signaling collapses, and the sender times out on waiting for any ACK packets from the receiver. Since the value of *cwnd* is below the *ssthresh* value, TCP also switches to *slow start* control mode, doubling the congestion window with every *RTT* interval.

It should be noted that the responsiveness of the TCP congestion avoidance algorithm is measured in intervals of the *RTT*, and the overall intent of the algorithm is to reach a steady state where the sender injects a new segment into the network at the same point in time when the receiver accepts a segment from the network.

It should be noted that this algorithm works most efficiently when the spacing between ACK packets as received by the sender matches the spacing between data packets as they were received at the remote end. It should also be noted that the algorithm works optimally when congestion-induced packet loss happens just prior to complete queue exhaustion. The intent is to signal packet loss through duplicate ACK packets, allowing the sender to undertake a fast retransmission and leave the congestion window at the *ssthresh* level. If queue exhaustion occurs, the TCP session will stop receiving ACK signals and retransmission will only occur after timeout, and at that point the congestion window is bought back to a single segment and the slow start algorithm is recommenced. An equal performance problem is network congestion events which occur within very short time intervals compared to the *RTT* (as may be the case in a mixed traffic load across a common ATM transport substrate, where short ATM switch cell queues may lead to very short interval cell loss events). In this case, a TCP session may switch from the aggressive window expansion of *slow start* into a much slower window expansion of *congestion avoidance* at a traffic rate well below the true long term network availability level.

One major drawback in any high-volume IP network is that when there are congestion hot spots, uncontrolled congestion can wreak havoc on the overall performance of the network to the point of congestion collapse. When a high volume of TCP flows are active at the same time, and a congestion situation occurs within the network at a particular bottleneck, each flow conceivably could experience loss at approximately the same time, creating what is known as *global synchronization. Global* refers to all TCP flows in a given network that traverse a common path. Global synchronization occurs when hundreds or thousands (or perhaps hundreds of thousands) of flows back off their transmission rates and revert to TCP slow start mode at roughly the same time. Each TCP sender detects loss and reacts accordingly, going into slow-start mode, shrinking its transmission window size, pausing for a moment, and then attempting to retransmit the data once again. If the congestion situation still exists, each TCP sender detects loss once again, and the process repeats itself over and over again, resulting in a network form of gridlock [6].

Uncontrolled congestion is detrimental to the network - behavior becomes unpredictable, system buffers fill up, packets ultimately are dropped, and the byproduct is a large number of retransmits which could ultimately result in complete congestion collapse.

3.3.2 Preferential Congestion Avoidance at Intermediate Nodes

Van Jacobson discussed the basic methods of implementing congestion avoidance in TCP in 1988 [7]. However, Jacobson's approach was more suited for a small number of TCP flows, which is much less complex to manage than the volume of active flows in the Internet today. In 1993, Sally Floyd and Van Jacobson documented the concept of RED (Random Early Detection), which provides a mechanism to avoid congestion collapse by randomly dropping packets from arbitrary flows in an effort to avoid the problem of global synchronization and, ultimately, congestion collapsed [8]. The principal goal of RED is to avoid a "queue tail drop" situation in which all TCP flows experience congestion at the same time, and subsequent packet loss, thus avoiding global synchronization. RED also attempts to create TCP congestion signals using duplicate ACK signaling, rather than through sender timeout, which in turn produces a less catastrophic rate backoff by TCP. RED monitors the mean queue depth, and as the queue begins to fill, it begins to randomly select individual TCP flows from which to drop packets, in order to signal the receiver to slow down [Figure 4]. The threshold at which RED begins to drop packets is generally configurable by the network administrator, as well as the rate at which drops occur in relation to how quickly the queue fills. The more it fills, the greater the number of flows selected, and the greater the number of packets dropped. This results in signaling a greater number of senders to slow down, thus resulting in a more manageable congestion avoidance.

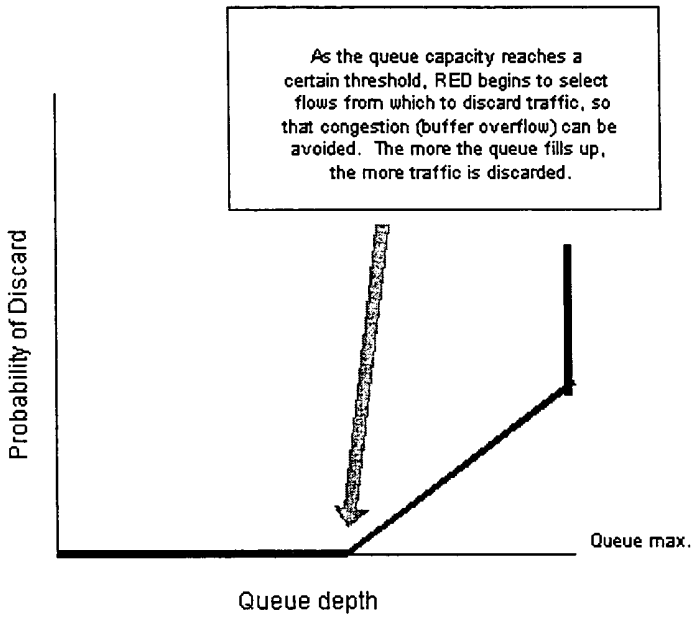


Figure 4

The RED approach does not possess the same undesirable overhead characteristics as some non-FIFO (First In, First Out) queuing techniques (e.g. simple priority queuing, class based queuing, weighted fair queuing). With RED, it is simply a matter of who gets into the queue in the first place - no packet reordering or queue management takes place. When packets are placed into the outbound queue, they are transmitted in the order in which they are queued. Queue-based scheduling mechanisms, such as priority, class-based, and weighted-fair queuing, however, require a significant amount of computational overhead due to packet reordering and queue management. RED requires much less overhead than these non-FIFO queuing mechanisms, but then again, RED performs a completely different function.

RED can be said to be fair - it chooses random flows from which to discard traffic in an effort to avoid global synchronization and congestion collapse, as well as to maintain equity in which traffic actually is discarded. Fairness is all well and good, but what is really needed for differentiated QoS structures is a tool that can induce unfairness - a tool that can allow the network administrator to predetermine what traffic is dropped first (or last, as the case may be) when RED starts to select flows from which to discard packets. You can't differentiate services with fairness.

There are several proposals in the IETF which have suggested using the IP precedence subfield of the TOS (Type of Service) byte contained in the IP packet header to indicate the relative priority, or discard preference, of packets and to indicate how

packets marked with these relative priorities should be treated within the network. As precedence is set or policed when traffic enters the network (at ingress), a weighted congestion avoidance mechanism implemented in the core routers determines which traffic should be discarded first when congestion is anticipated due to queue-depth capacity. The higher the precedence indicated in a packet, the lower the probability of discard. The lower the precedence, the higher the probability of discard. When the congestion avoidance is not actively discarding packets, all traffic is forwarded with equity.

Of course, for this type of operation to work properly, an intelligent congestion-control mechanism must be implemented on each router in the transit path. A least one currently deployed mechanism is available that provides an unfair, or weighted, behavior for RED. This deviation of RED yields the desired result for differentiated traffic discard in times of congestion and is called *Weighted Random Early Detection* (WRED). A similar scheme, called *enhanced RED*, is documented in a paper authored by Feng, Kandlur, Saha, and Shin [9].

3.3.3 Scheduling Algorithms to Implement Differentiated Service

There are other approaches to implement differential QoS within the network which use the routers queuing algorithm (or *scheduling discipline*) as the enabling mechanism. Considering that queuing is perhaps the optimal point to introduce QoS differentiation mechanisms this is an area of considerable interest.

FIFO Queuing

The base of best effort, single quality, network environments is that of a FIFO queue, where there is no inherent differentiation undertaken by the router's transmission scheduler. Every packet which is scheduled to be transmitted on an output interface must await all previously scheduled packets before transmission. All such packets occupy slots in a single per-interface queue, and when the queue fills, all subsequent packets are discarded until the queue becomes available once more. As with the basic RED algorithm, this is a fair algorithm, given that it allocates the transmission resource fairly and imposes the same delay on all queued packets. For differentiated service levels, it is necessary to alter this fairness and introduce mechanisms to trigger preferential outcomes for classes of traffic.

The basic modification of the single level FIFO algorithm to enable differentiated QoS is to divide traffic into a number of categories, and then provide resources to each category in accordance with a predetermined allocation structure, implementing some form of proportional resource allocation.

Of course, the *Law of Conservation* holds here, such that the sum of the mean queuing delays per traffic category, weighted by their share of the resources they receive is limited, with the corollary that in reducing the mean queuing delay for one category of traffic will result in the increase in mean queuing delay for one or more of the remaining categories of traffic [10]. Accordingly, you can't improve the performance profile of one class of traffic without adversely affecting the performance profile of one or more of the other classes of traffic, and the level of degradation will be similar in quantity to the level of improvement that was effected.

Priority Queues

A basic modification to the base FIFO structure is to create a number of distinct queues for each interface and associate a relative priority level to each. Packets are scheduled from a particular priority queue in FIFO order only when all queues of a higher priority are empty. In such a model, the highest priority traffic receives minimal delay, but all other priority levels may experience resource starvation if the highest precedence traffic queue remains occupied. To ensure that all traffic receives some level of service, it is a requirement that the network uses admission policies which restricts the amount of traffic which is admitted at each elevated priority, or that the scheduling algorithm is adjusted to ensure that every priority class receives some minimum level of resource allocation. Accordingly, this simple priority mechanism does not scale well, although it can be implemented with relatively little cost, and more sophisticated (and more robust) scheduling algorithms are required within the Internet for QoS support.

Generalized Processor Sharing

The ideal approach is to associate a relative weight (or precedence) with each individual traffic flow, and at every router, segment each traffic flow into an individual FIFO queue, and configure the scheduler to service all queues in a bit-wise round robin fashion, allocating service to each flow in accordance with the relative weight. This is an instance of a *Generalized Processor Sharing* (*GPS*) discipline [11 - pp. 234-236].

Weighted Round-Robin and Deficit Weighted Round Robin

Various scheduling techniques can approximate this model. A basic approach is to use a packet's marked precedence to place the packet into a precedence-based queue, and then use a *weighted round-robin* scheduling algorithm to service each queue. If all packets are identically sized, this is a relatively good approximation of *GPS*, but when packet sizes vary, this algorithm can exhibit significant deviation from a strict relative resource allocation strategy. This can be partially addressed using a *deficit weighted round-robin* [11 - pp. 237-238] algorithm which modifies the round robin algorithm to use a service quantum unit. A packet is scheduled from the head of a weighted queue only if the packet size minus the per-queue deficit counter is less than the weighted quantum value, and the next packet in the queue is tested using a weighted quantum value which has been reduced by the size of the scheduled packet. When the test fails, the remaining weighted quantum size is added to the per-queue deficit counter and the scheduler moves to the next queue. This algorithm performs with an average allocation which corresponds to the relative weights of each queue, but still exhibits unfairness within time frames which are commensurate to the maximum packet service time.

Weighted Fair Queuing

Weighted Fair Queuing (*WFQ*) [12] attempts to provide fairer resource allocation measures which protect "well behaved" traffic sources from uncontrolled sources. *WFQ* attempts to compute the finish time of each queued packet if a bit-wise weighted *GPS* scheduler had been used, and then schedules for service the packet with the smallest finish time which would've been receiving service in the corresponding GPS scheduler model. *WFQ* is both a scheduling and packet drop policy, where packet drop is based on a preference for dropping packets with the greatest finish time in response to an incoming packet which requires a queue slot. While *WFQ* requires a relatively complex implementation, it has a number of desirable properties. The scheduling algorithm does undertake fair allocation which does indeed ensure that different categories of traffic are not capable of resource starving other categories. The algorithm also bounds the queue delay per service category, which also provides a lever to create delay-bounded services without the need for resource reservation.

3.3.4 Traffic Shaping Non-Adaptive Flows

One of the more confounding aspects of providing differentiated services at the network and transport layers of the TCP/IP protocol suite is that of dealing with non-adaptive flows, or in other words, traffic flows which do not adapt their transmission rates in response to loss in the network. The most offensive of this category appear to be applications which use UDP as their transport protocol. This is somewhat ironic in that long-standing traditional thinking has assumed that applications which use UDP are generally thought to be designed to be "intelligent enough" to recognize loss, since UDP does not provide any error correction itself. The resulting observation is that this is a fundamentally flawed assumption, since it is generally recognized that applications which use UDP are generally not very "network friendly."

Having said that, the subsequent action is to rate-shape UDP flows as they enter the network, limiting their transmission rate to a specified bit rate. This method is arguably a compromise - it's not pretty, but we understand how to do this, and it works. There are, however, a couple of proposed methods to enhance the basic RED mechanism to provide some relief in the face of non-adaptive flows, however, the validity and practicality of these schemes are still being evaluated. One such proposal is discussed in [13].

3.4 Integrated Services and RSVP

It has been suggested that the Integrated Services architecture [14] and RSVP [15] are excessively complex and possess poor scaling properties. This suggestion is undoubtedly prompted by the existence of the underlying complexity of the IP layer signaling requirements. However, it also can be suggested that RSVP is no more complex than some of the more advanced routing protocols. An alternative viewpoint might suggest that the underlying complexity is required because of the inherent difficulty in establishing and maintaining path and reservation state information along the transit path of data traffic. The suggestion that RSVP has poor scaling properties deserves additional examination, however, because deployment of RSVP has not been widespread enough to determine the scope of this assumption.

As discussed in [16], there are several areas of concern about the wide-scale deployment of RSVP. With regard to concerns of RSVP scalability, the resource requirements (computational processing and memory consumption) for implementing RSVP on routers increase in direct proportion to the number of separate RSVP reservations, or sessions, accommodated. Therefore, supporting a large number of RSVP reservations could introduce a significant negative impact on router performance. By the same token, router-forwarding performance may be impacted adversely by the packet classification and scheduling mechanisms intended to provide differentiated services for reserved flows. These scaling concerns tend to suggest

that organizations with large, high-speed networks will be reluctant to deploy RSVP in the foreseeable future, at least until these concerns are addressed. The underlying implications of this concern also suggest that without deployment by Internet service providers, who own and maintain the high-speed backbone networks in the Internet, the deployment of pervasive RSVP services in the Internet will not be forthcoming.

Another important concern expressed in [16] deals with policy-control issues and RSVP. Policy control addresses the issue of who is authorized to make reservations, and encompasses provisions to support access control and accounting. Although the current RSVP specification defines a mechanism for transporting policy information, it does not define the policies themselves, because the policy object is treated as an opaque element. Some vendors have indicated that they will use this policy object to provide proprietary mechanisms for policy control. At the time of this writing, however, the IETF has chartered a new working group, called the RSVP Admission Policy (rap) working group [17], to develop a simple policy-control mechanism to be used in conjunction with RSVP. There is ongoing work on this issue in this working group. Several mechanisms already have been proposed to deal with policy issues, however, it is unclear at this time whether any of these proposals will be implemented or adopted as a standard.

The key recommendation contained in [16] is that given the current form of the RSVP specification, multimedia applications run within smaller, private networks are the most likely to benefit from the deployment of RSVP. The inadequacies of RSVP scaling, and lack of policy control, may be more manageable within the confines of a smaller, more controlled network environment than in the expanse of the global Internet. It certainly is possible that RSVP may provide genuine value and find legitimate deployment utility in smaller networks, both in the peripheral Internet networks and in the private arena, where these issues of scale are far less critical. Therein lies the key to successfully delivering QoS using RSVP. After all, the purpose of the Integrated Services architecture and RSVP is to provide a method to offer quality of service, not to degrade the service quality.

4.0 Conclusions

A number of dichotomies exist within the Internet that tend to dominate efforts to engineer possible solutions to the quality of service requirement. Thus far, QoS has been viewed as a wide-ranging solution set against a very broad problem area. This fact often can be considered a liability. Ongoing efforts to provide "perfect" solutions have illustrated that attempts to solve all possible problems result in technologies that are far too complex, have poor scaling properties, or simply do not integrate well into the diversity of the Internet. By the same token, and by close examination of the issues and technologies available, some very clever mechanisms are revealed under close scrutiny. Determining the usefulness of these mechanisms, however, is perhaps the most challenging aspect in assessing the merit of any particular QoS approach.

In the global Internet, however, it becomes an issue of implementing QoS within the most common denominator - this is clearly the TCP/IP protocol suite - because a single link-layer media will never be used pervasively end-to-end across all possible paths. What about the suggestion that it is certainly possible to construct a smaller network of a pervasive link-layer technology, such as ATM? Although this is certainly possible in smaller private networks, and perhaps in smaller peripheral networks in the Internet, it is rarely the case that all end-systems are ATM-attached, and this does not appear to be a likely outcome in the coming years. In terms of implementing visibly differentiated services based on a quality metric, using ATM only on parts of the end-to-end path is not a viable answer. The ATM subpath is not aware of the complete network layer path, and it does not participate in the network or transport layer protocol end-to-end signaling.

The simplistic answer to this conundrum is to dispense with TCP/IP and run native cell-based applications from ATM-attached end-systems. This is certainly not a realistic approach in the Internet, though, and chances are that it is not very realistic in a smaller corporate network, either. Very little application support exists for native ATM. Of course, in theory, the same could have been said of Frame Relay transport technologies in the recent past, and undoubtedly will be claimed of forthcoming transport technologies in the future. In general, link layer technologies are similar to viewing the world through plumber's glasses - every communications issue is seen in terms of point-to-point bit pipes. Each wave of transport technology attempts to add more features to the shape of the pipe, but the underlying architecture is a constant perception of the communications world as a set of one-on-one conversations, with each conversation supported by a form of singular communications channel.

One of the major enduring aspects of the communications industry is that no such thing as a ubiquitous single link layer technology. Hence, there is an enduring need for an internetworking end-to-end transport technology that can straddle a heterogeneous link layer substrate. Equally, there is a need for an internetworking technology that can allow differing models of communications, including fragmentary transfer, unidirectional data movement, multicast traffic, and adaptive data flow management.

This is not to say that ATM itself, or any other link layer technology for that matter, is not an appropriate technology to install into a network. Surely, ATM offers high-speed transport services, as well as the convenience of virtual circuits. However, what is perhaps more appropriate to consider is that any particular link layer technology is not effective insofar as providing QoS in the Internet for reasons that have been discussed thus far.

To quote a work in progress from the Internet Research Task Force, "The advantages of [the Internet Protocol's] connectionless design, flexibility and robustness, have been amply demonstrated. However, these advantages are not without cost - careful design is required to provide good service under heavy load." [18]. Careful design is not exclusively the domain of the end-system's protocol stack, although good end-system stacks are of significant benefit. Careful design also includes consideration of the mechanisms within the routers that are intended to avoid congestion collapse. Differentiation of services places further demands on this design, because in attempting to allocate additional resources to certain classes of traffic, it is essential to ensure that the use of resources remains efficient, and that no class of traffic is totally starved of resources to the extent that it suffers throughput and efficiency collapse.

For QoS to be functional, it appears to be necessary that all the nodes in a given path behave in a similar fashion with respect to QoS parameters, or at the very least, do not impose additional QoS penalties other than conventional best effort into the end-to-end traffic environment. The sender (or network ingress point) must be able to create some form of signal associated with the data that can be used by downstream routers to potentially modify their default outbound interface selection, queuing behavior, and/or discard behavior.

The insidious issue here is attempting to exert "control at a distance." The objective in this QoS methodology is for an end-system to generate a packet that can trigger a differentiated handling of the packet by each node in the traffic path, so that the end-to-end behavior exhibits performance levels in line with the end-user's expectations and perhaps even a contracted fee structure.

This *control-at-a-distance* model can take the form of a "guarantee" between the user and the network. This guarantee is one in which, if the ingress traffic conforms to a certain profile, the egress traffic maintains that profile state, and the network does not distort the desired characteristics of the end-to-end traffic expected by the requestor. To provide such absolute guarantees, the network must maintain a transitive state along a determined path, where the first router commits resources to honor the traffic profile and passes this commitment along to a neighboring router that is closer to the nominated destination and also capable of committing to honor the same traffic profile. This is done on a hop-by-hop basis along the transit path between the sender and receiver, and yet again from receiver back to sender. This type of state maintenance is viable within small-scale networks, but in the heart of large-scale public networks such as the global Internet, the cost of state maintenance is overwhelming. Because this is the mode of operation of RSVP, this presents some serious scaling considerations and is inappropriate for deployment in large networks.

RSVP scaling considerations present another important point, however. RSVP's deployment constraints are not limited simply to the amount of resources it might consume on each network node as per-flow state maintenance is performed. It is easy to understand that as the number of discrete flows increases in the network, the more resources it will consume. Of course, this can be somewhat limited by defining how much of the network's resources are available to RSVP - everything in excess of this value is treated as best-effort. What is more subtle, however, is that when all available RSVP resources are consumed, all further requests for QoS are rejected until RSVP-allocated resources are released. This is similar in functionality to how the telephone system works, where the network's response to a flow request is commitment or denial, and such a service does not prove to be a viable method to operate a data network where better-than-best-effort services arguably should always be available.

The alternative to state maintenance and resource reservation schemes is the use of mechanisms for preferential allocation of resources, essentially creating varying levels of best-effort. Given the absence of end-to-end guarantees of traffic flows, this removes the criteria for absolute state maintenance, so that "better-than-best-effort" traffic with classes of distinction can be constructed inside larger networks. Currently, the most promising direction for such better-than-best-effort systems appears to lie within the area of modifying the network layer queuing and discard algorithms. These mechanisms rely on an attribute value within the IP packet's header, so these queuing and discard preferences can be made at each intermediate node. First, the ISP's routers must be configured to handle packets based on their IP precedence level, or similar semantics expressed by the bit values defined in the IP packet header. There are three aspects to this. First, you need to consider the aspect of using the IP precedence field to determine the queuing behavior of the router, both in queuing the packet to the forwarding process and queuing the packet to the output interface. Second, consider using the IP precedence field to bias the packet discard processes by selecting the lowest precedence packets to discard first. Third, consider using any priority scheme used at Layer 2 that should be mapped to a particular IP precedence value.

There are several methods have been proposed within the IETF which may yield robust mechanisms and semantics for providing these types of differential services (*diffserv*) [19].

The generic *diffserv* deployment environment assumes that the network uses ingress traffic policing, where traffic passed into the network is passed through traffic shaping profile mechanisms, which bound their average and peak data rates, and their relative priority and discard precedence in accordance with the traffic profile and the administrative agreement with the customer. These ingress filters can be configured to either discard out-of-profile packets, or the ingress filter may mark them with an elevated discard priority so that they are carried within the network only when there are adequate resources available. Within the core of the network, *WFQ* (or similar proportional scheduling algorithms) can be used to allocate network resources according to the marked priority levels, allowing the high speed and high volume switching component of the network to operate without per-flow state being imposed.

The cumulative behavior of such stateless, local-context algorithms and corresponding deployment architectures can yield the capability of distinguished and predictable service levels, and hold the promise of excellent scalability. You still can mix best-effort and "better-than-best-effort" nodes, but all nodes in the latter class should conform to the entire QoS selected profile or a compatible subset (an example of the principle is that it is better to do nothing than to do damage).

In conclusion, QoS is possible in the Internet, but it does come at a price of compromise - there are no perfect solutions. In fact, one might suggest that expectations have not been appropriately managed, since guarantees are simply not possible in the Internet, at least not for the foreseeable future. What is possible, however, is delivering differentiated levels of best effort traffic in a manner which is predictable, fairly consistent, and which provides the ability to offer discriminated service levels to different customers and to different applications.

5.0 References

[1] IETF Internet Draft, "End-to-End Traffic Issues in IP/ATM Internetworks," draft-jagan-e2e-traf-mgmt-00.txt, S. Jagannath, S. Yin, August 1997.

[2] "MAC Bridges," ISO/IEC 10038, ANSI/IEEE Std. 802.1D, 1993.

[3] "Supplement to MAC Bridges: Traffic Class Expediting and Dynamic Multicast Filtering," IEEE P802.1p/D6, May 1997.

[4] IETF Internet Draft, "Integrated Service Mappings on IEEE 802 Networks," draft-ietf-issll-is802-svc-mapping-01.txt, M. Seaman, A. Smith, E. Crawley, November 1997.

[5] IETF Internet Draft, "SBM (Subnet Bandwidth Manager): A Proposal for Admission Control over IEEE 802-style networks," draft-ietf-issll-is802-bm-05.txt, R. Yavatkar, F. Baker, D. Hoffman, Y. Bernet, November 1997.

[6] "Oscillating Behavior of Network Traffic: A Case Study Simulation," L. Zhang, D. Clark, Internetwork: Research and Experience, Volume 1, Number 2, John Wiley & Sons, 1990, pages 101-112.

[7] "Congestion Avoidance and Control," V. Jacobson, Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.

[8] "Random Early Detection Gateways for Congestion Avoidance," S. Floyd, V. Jacobson, IEEE/ACM Transactions on Networking, v.1, n.4, August 1993.

[9] "Understanding TCP Dynamics in an Integrated Services Internet," W. Feng, D. Kandlur, D. Saha, K. Shin, NOSSDAV '97, May 1997.

[10] "Queuing Systems, Volume 2: Computer Applications," L. Kleinrock, Wiley Interscience, 1975.

[11] "An Engineering Approach to Computer Networking," S. Keshav, Addison-Wesley, 1997.

[12] "Design and Analysis of a Fair Queuing Algorithm," A. Demera, S. Keshav, and S. Shenker, ACM SIGCOMM'89, Austin, September 1989.

[13] "Dynamics of Random Early Detection," D. Lin and R. Morris (Harvard University), a proposal & overview of Fair Random Early Drop (FRED). Presented at ACM SIGCOMM 1997.

[14] RFC1633, "Integrated Services in the Internet Architecture: An Overview," R. Braden, D. Clark, S. Shenker, June 1994.

[15] RFC2205, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification," R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, September 1997.

[16] RFC2208, "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement, Some Guidelines on Deployment," A. Mankin, F. Baker, R. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, L. Zhang, September 1997.

[17] The IETF RSVP Admission Policy (rap) working group charter can be found at: http://www.ietf.org/html.charters/rap-charter.html

[18] Internet Research Task Force draft, "Recommendations on Queue Management and Congestion Avoidance in the Internet," draft-irtf-e2e-queue-mgt-recs.ps, R. Braden, D. Clark, J. Crowcroft, B. Davie, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, March 1997.

[19] Differential Service for the Internet, http://diffserv.lcs.mit.edu/

AGERE White Paper

Building Next Generation Network Processors

Agere, Inc.

Abstract

RISC processor architectures have been exceptionally successful in solving network-processing challenges to date. As network interfaces have exceeded gigabit rates, however, sequential RISC processor bus speeds and clock speeds have not typically scaled to keep up. Further, as required speeds increase, so do the expectations of what must be done with this data at line rates.

Network equipment vendors have deployed specialized hardware to meet their line rate scaling needs where software based, sequential processor solutions would not suffice. These devices tend to have limited programmability, resulting in long turn around times to spin next generation silicon as feature and function requirements evolve.

It is clear that the next generation of network processors employing a new programmable architecture is needed to scale to a gigabit and beyond. This paper describes the limitations of RISC-based architectures operating in this environment. It details the requirements of a new architecture that can deal with these limitations and the techniques that a new architecture may employ, such as pipelining, parallelism, and separate program and I/O data paths.

Network Processing Functions

To better describe the new network processor topology, its important to understand the underlying functions in network processing. These functions can be divided into four categories. *Pattern Matching, Data Manipulation, Queue Management*, and *Statistics Gathering.*

Pattern Matching

Pattern Matching ranges from bit-field associations to forwarding on destination address. The former is easily done in software using a case statement or by bit-wise decoding in hardware. Destination address forwarding, which involves larger bit-field lookups, requires different techniques depending on the size of the address field. In the case of ATM or MPLS, where the address fields are relatively small, a single lookup to memory can obtain a result. For larger address fields, significant amounts of memory are required. For example, a direct memory lookup for the IPv4[1] address space returning a 16-bit value requires 8 Gigabytes of memory.

In the cases of larger address ranges, multiple lookup operations are normally required. The compute intensity of larger tree lookups has spawned lookup techniques which are optimized for either minimal memory operations (speed) or minimal memory required (size) to store a lookup table.[2] Given that a subset of addresses are actually being serviced in a given node in a network, algorithms which optimize memory utilization and I/O operations have been devised to make optimal use of memory and memory operations.

Data Manipulation

The incoming and outgoing bit stream generally requires some modification as it is processed. These functions range significantly in complexity. Fields such as the Time To Live (TTL) field in an IP header need to be checked for viability and decremented as required, and any CRC fields need to be recalculated as a result of such changes.

Adding bit fields to a data packet adds a different range of complexity. In some instances, a tag or label is appended (or prepended) to the packet. In some instances the data is encapsulated into entirely new packets. A VPN application might encrypt the entire packet and encapsulate it as a payload in a

[1] Internet Protocol Version 4 utilizes a 32 bit wide address field to identify the destination of a packet.

[2] "Small Forwarding Tables for Fast Lookups" by Degermark, Broadnik, Carlsson, Pink of Lulea University of Technology, Sweden; "High Speed Routing Lookups" by Valdvogel, and Plattner of Compter Networking Laboratory of Zurich, Switzerland and Norghese and Turner of Washington University.

new packet with a new header.

Packet segmentation, re-assembly, and fragmentation also fall under this class of functions. This type of processing requires state information to be maintained for each stream of packets or cells being processed.

Queue and Buffer Management

A variety of network functions fall under the generic category of Queue Management. These functions range from packet assembly and segmentation to Quality of Service queuing, where ingress data is re-ordered at the egress queue by priority. Data discard and traffic shaping algorithms also come under this classification.

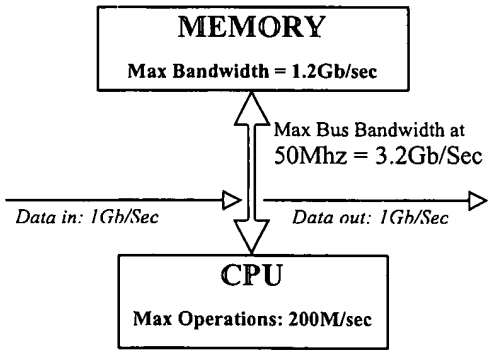
Such functions require buffering, where network data is stored into logical queues, and a decision engine which determines the rate, priority, and type of servicing required for each queue.

Statistics Gathering

From a processing viewpoint, statistics in a network environment divide into data that can be gathered in the background, and statistics that must be done on a per-packet basis. Background statistical data can often be gathered via polling or interrupt-based processing; packet rate statistics generally require a search on a portion of the incoming data and incrementing the appropriate counter(s).

RISC SCALING

Clearly there are options to scale RISC to handle higher data speeds and more complex processing. The question is at what point do alternative architectures become more optimal and attractive solutions? To answer this question, a review of how RISC can be scaled and limitations of each technique is required. First however, let's look at an example of a single RISC processor handling single gigabit interface.



In this example, a one gigabit/second IP data stream is coming in on a single port. Assuming an average packet size of 100 bytes, the processor must forward approximately 1.25M packets/second on an average. The major performance bottlenecks of concern are bus bandwidth requirements to handle data traffic along with the total number of processor clock cycles required to receive packet data, perform a forwarding table lookup, perform any data modifications required, and transmit the result. In the example above, a processor with a bus width of 64 bits that clocks at 50Mhz has a peak bus bandwidth of 3.2Gb/second. Assuming all packet data goes into and out of the processor, and a data rate of 1Gb/second, means 2Gb/second of the bus bandwidth is taken up solely by ingress and egress line data. This leaves 1.2Gb/second available for other bus operations and equates to 18.75 million bus operations per second of overhead. Divide this by 1.25M Packets/second to determine a bus operation average/packet and the result is 15 bus transfers of 64bits per packet! Note also that this is an unrealistically optimistic number given that it assumes that the processor can schedule bus operations with 100% efficiency and doesn't require bus time to perform operating system tasks and cache fills.

Incoming Packet Rate

= Bit Rate ÷ Bytes per Packet ÷ 8 Bits per Byte

= 1 Gbits/sec ÷ 100 bytes/packet ÷ 8 bits/byte

= 1.25 M packets/sec

Available Bus

= Max Bus Bandwidth – Ingress – Egress

= 3.2 Gbits/sec – 1 Gbits/sec – 1 Gbits/sec

= 1.2 Gbits/sec

Max Bus Operations Per Packet

= Available Bus ÷ Bus Size ÷ Incoming Packet Rate

= 1.2 Gbits/sec ÷ 64 bits ÷ 1.25 M packets/sec

= 15 /packet

Next, if we look at the processors internal operations, assuming the CPU can perform one internal operation every 200Mhz clock cycle this equates to 160 instructions available per packet processed. This is also an unrealistically optimistic number since the CPU is unlikely to be able to maintain a 100% utilization of its instruction engine. Given the complex instructions required to process an entire IP packet plus operating system overhead, 160 instructions per packet appears to be significantly less than what would be required for the job from a data

processing standpoint.[3]

Max CPU Operations per Packet

= CPU Speed × Ops per Cycle + Packet Rate

= 200 MHz × 1 ops/cycle + 1.25 M packet/sec

= 160 ops/packet

This example clearly shows that a relatively state-of-the-art RISC processor cannot handle a 1 gigabit data stream from the standpoint of bus performance or data processing performance. To mitigate this problem, many scaling techniques have been employed.

RISC Scaling Techniques

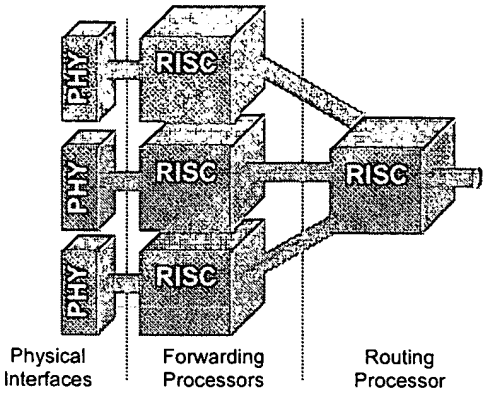
Function Partitioning

Separating out the processing functions in network equipment and delegating them to dedicated processors is the most obvious step to scaling a sequential processor architecture. Functions such as forwarding and routing/signaling can be partitioned and delegated to different processors. In IP/ATM applications Segmentation and Reassembly of a packet from ATM cells is a common function that is partitioned out to a separate processor.

Parallel Processing

Multiple processors can be used to alleviate performance issues in several ways. The most stable and common technique is to partition system functions and dedicate a processor to a function. In IP routers, for example, it is becoming increasingly common to separate the Forwarding function from the Routing function. To keep the processing requirements within the means of a RISC, forwarding is often distributed on a per interface basis, and a single RISC processor handles the forwarding on a single interface (or a subgroup of interfaces).

This form of parallelism scales nicely up to the point where a single interface's data rate exceeds the processor's I/O or processing capabilities. The breaking point from a complexity standpoint is where a single processor can no longer handle a single interface. As more parallelism is deployed on a single stream of data, maintaining context between processors, and synchronization of tasks become more and more complex. The software model gets increasingly complex as well. Any changes in functionality causes a further strain on the system as more parallel processors are added and processes are re-distributed between them.



Multiple RISC processors can be applied to different functions to scale the effectiveness of the RISC architecture.

Cache Optimization

The challenges with a cache-centric processor doing I/O intensive operations revolve around reserving as much bus bandwidth for I/O as possible and optimizing performance by leveraging the processors internal cache. Optimizing the software such that the entire instruction memory fits in the processor's instruction cache eliminates cache misses from instruction fetches, once all instructions have been accessed once. Forwarding algorithms have been optimized in the effort to achieve a forwarding table that fit's in a processor's internal cache, or the internal cache can be sized to meet the forwarding requirements.

Bus Optimization

Reducing the amount of network data required by a processor is one technique used to scale a processor's I/O capabilities. For example, if only pertinent header information is required by a processor doing forwarding, then there is a significant performance gain especially large average packet sizes in a network. This technique was used at BBN in the DARPA MGR program.[4]

Network Function Acceleration Co-processors

More complex line rate functions are increasingly delegated to dedicated co-processors. ATM Seg-

[3] "RISC" refers to a reduced instruction set. This means that fewer, more general instructions are available than are available with other architectures such as CISC. This normally means that more instructions are necessary to achieve a result.

[4] "A 50-Gb/s Router" by Partridge and Carvey of BBN Systems & Technology.

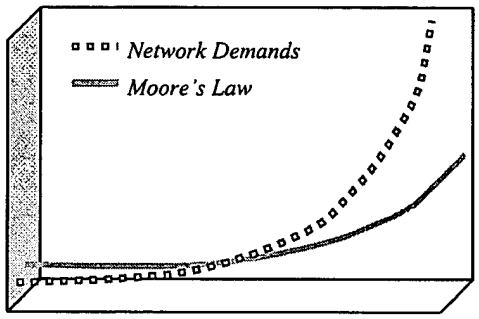
mentation and Reassembly is generally done this way today. Many link layer functions such as HDLC for PPP sessions delegated to dedicated hardware state machines.

Scaling Limits

Each of the techniques described above has particular boundaries, beyond which, scaling is not practical.

Clock & Bus Speed Scaling

Given the increase in interface speeds and performance demands as compared to the rate at which RISC clock and bus speeds are increasing, it's clear that this architecture is not going to keep up with network demands.



Network speed and bandwidth demands are dramatically out-pacing advancements in silicon processing.

Parallelism and Pipelining

Pipelining techniques used in sequential processors are optimized for general purpose processing. Data fetch, instruction fetch, and instruction processing are pipelined in a manner such that given a range of operations, resources utilization is optimized on a statistical basis. This model works well given a certain degree of randomness in the kinds of instructions and operations being sequenced through the processor. Fixed patterns of processing in certain instances may not make optimal use of bus or processing resources. Caching is used to offset some of these limitations (specifically bus utilization); however, as explained later, caching can be a barrier to I/O intensive operations. To make use of the pipelined architectures in a RISC, requires fairly low-level program-level optimizations. This complicates matters when new features need to be implemented and the program needs to be fine-tuned at a low level to meet performance demands.

Adding parallel processors to the equation may improve performance, but with diminishing results as parallelism increases. Reasons for the diminishing returns on performance include overhead of performing such tasks as maintaining context between processors, merging results, etc.

Cache Centric Processing

Caching has been implemented in RISC processors to optimize performance given that the processor clocks internally external I/O bus. Caching techniques used in general purpose RISC processors however are designed to produce performance enhancement for general purpose computing and not necessarily for I/O intensive operations.

In an I/O intensive network application, a general purpose, sequential processor caching scheme generally does not provide the same performance advantages. The bus overhead for many caching implementations can provide a significant hindrance to the task of getting large quantities of I/O data on and off chip.

In a network processing applications, the data that is consistently re-accessed is generally table lookup data. Rather than a cache, implementing a fast internal SRAM is more efficient.

While caching can provide measurable performance gains in a network processor, these techniques must be applied with the specific applications in mind.

Embedding RISC Cores into ASICs

Increasingly ASIC vendors have made RISC cores available as a standard logic block to be implemented into custom chip designs. This has made it possible to add network specific custom hardware to perform processing intensive tasks like Lookups and ATM Segmentation and Reassembly. Some customization of these cores can be deployed to improve context switching performance, I/O and control data path performance, etc. Too much modification, however, requires modification of software development tools. Additionally, the question must be asked, given that these processor cores have been optimized to perform well in a very different context from I/O intensive networking operations, how much precious gate overhead is being taken up by circuitry which has no real use to the application? Adding multiple cores in an ASIC only multiplies the un-necessary gate overhead.

Programming Complexity

The increasing complexity of a parallel RISC programming model, especially when non-standard acceleration hardware has been added, creates a programming environment that has significant scaling limits. Both performance and feature enhancement activities become a major challenge for

a programmer and increasingly requires extensive knowledge of the internal workings of the processor's architecture in order to achieve any kind of result.

The End of RISC Processor scaling

It becomes clear after reviewing all of the factors above, that to continue to add optimizations to a sequential, cache-centric processor will reach a point of diminishing returns quickly in the Network Processing arena. A new processing architecture is needed which gets clock speeds and gate counts back into the range where the designer has headroom for future scaling needs. Programmability becomes a major issue to factor in as well.

When lookup tasks are examined in the context of a RISC processor, it's sequential operations require multiple clock cycles to retrieve the network data and then perform a sequence of table look up and compare operations to obtain a result. Content Addressable Memory (CAM), attached to a RISC to reduce this overhead, still lacks scalability in the long run, considering the size of lookup tables and the number of lookups required in network search operations. In reality, a pipelined approach needs to be adopted.

Content Addressable Memories

CAMs continue to be useful tools for network data processing. They can significantly reduce the sequential operations required by a processor performing table lookups. Hybrid CAM search engines can indeed be useful special function lookup blocks for various networking search functions. CAM technology is generally not a viable solution where a general purpose, highly programmable lookup engine is required. Programming is difficult and performance of a CAM lookup engine is data dependent.

Hash Tables

Hashing has been deployed both in hardware and in software on conventional processors as a means to reduce the number of memory accesses to achieve a lookup result. A variety of hashing techniques exist. All are generally data dependent. When collisions occur, exception processing is required.

Alternative Processor Architectures

Pipelining

When Digital Signal Processors (DSPs) became available for I/O intensive signal processing applications, they were quickly embraced due to the significant performance improvements that were possible in those applications. The Harvard Architecture (separate Program and Data memory & busses), and it's derivatives made sense for I/O intensive activities. This provided a separate bus for I/O operations which could be scaled independently according to the signal bandwidth requirements, eliminating the non-deterministic behavior that arises when this bus is shared by program and data space. Pipelining multiple processing blocks was also an optimal solution given that a constant stream of signal data is expected in signal processing applications. While the pipeline added latency, eliminating the need for multiple sequential operations from a single processing element on a single data entity meant that once the pipeline was full, a result could be had at the completion of every processor clock cycle.

For significantly less aggressive processor clock speeds, in the applications DSPs were designed for, appreciable performance could be realized.

These very techniques are exceedingly apropos in network data processing applications. Where DSP's themselves fall short in such applications is the fact that they have been optimized to perform fast forier transforms efficiently and the processing operations for this particular operation do not map well to networking applications. Additionally, DSP's are notoriously difficult to program in a higher level programming language. Extensive knowledge of the inner workings of a DSP is necessary, if not required, for a programmer to make efficient use of the processing resources.

The Next Generation Network Processor

Pipelining and parallelism are essential performance enhancing techniques for processor design. Both techniques are used in sequential processors to enhance performance.

Processor function blocks optimized for network processing operations are also useful, however for a re-useable, multi-function processor, the functions where these blocks are applied must be carefully chosen for programmability and re-usability across a variety of network implementations, if the processor is to be successful as an off the shelf solution.

To determine the ideal network processor architecture to meet future programmability and performance needs, an optimal approach is to look at the

various network processing functions and determine from the pallete of processor techniques which techniques work best for each function.

The pallete to choose from includes:

- Sequential Processing
- Pipelined Processing
- Parallelism
- Specialized Function Blocks
- I/O optimization

Sequential Processing

RISC and CISC architectures have their unique advantages, however, given the assumption that parallel blocks will be used, gate count and complexity challenges will exist. This tends to favor a RISC architecture. An additional concern regarding Sequential Processing is that simple functions take multiple sequential operations to achieve a result. This equates to multiple clock cycles. The clock speed required of a RISC or CISC implementation must be examined from a stand point of future scalability. If state-of-the-art clock speeds are required of a Sequential Processor core to meet current performance requirements, can the state of the art keep up with future performance needs for the given application? Also, the efficiency of the instruction set must be taken into account.

Pipelined Processing

Many sequential processing architectures utilize pipelining techniques to enhance their performance. Specifically, instruction and data fetches have been pipelined to improve I/O performance. An entirely pipelined processor takes an assembly line approach to a problem which requires lower clock rates to achieve similar functionality in a sequential approach. Latency of the pipeline must be taken into account in a pipelined approach. Making a pipelined processor programmable creates challenges for the programmer. Functions used commonly in sequential processors, such as conditional branches, have the potential of rendering the results of one or more stage in the pipeline obsolete. If results of one or more pipestage must be discarded, performance of the processor is degraded.

Pipelined engines which are carefully designed for their intended applications, and are carefully programmed, can achieve optimal processing throughput at lower clock speeds and generally lower hardware gate counts than a Sequential processor counterpart.

Specialized Function Blocks

Sequential and Pipelined engines can achieve performance gains by utilizing specialized function blocks to perform application-specific processing functions. Special function logic units can often perform mathematical calculations or table lookup functions in significantly fewer clock cycles than when implemented in a sequential processor utilizing its instruction set. Specialized function blocks can also be deployed in a pipelined fashion with the objective of reducing the quantity of pipeline stages. A hardware based N x M multiply block could, for instance, be deployed in a Sequential processor with a special multiply instruction, or in a pipeline engine as a pipe stage in the processor.

Parallel Processing

Combinations of Pipelined, Sequential and specialized processor blocks could be used in parallel as well. Specialized function blocks can be deployed as a co-processor where a processing task can be handed off to be performed in parallel. Tasks that lend themselves to pipelined processing could be handled in parallel with tasks which require a more sequential approach. Parallel processing hurdles that must be resolved are:

1. Synchronizing and aggregating parallel processed threads.
2. Partitioning of processing tasks for parallel processing
3. Maintaining context and sharing of interim results between parallel blocks

The more independent each parallel stage is from each other, the fewer challenges there are in maintaining context or aggregating results.

I/O Optimization

A key area of concern for network processing is I/O operations. Servicing a high-bandwidth data stream coupled with memory bandwidth requirements for such I/O intensive processing require different bus architectures than those used in a shared bus sequential processor model.

The obvious optimization is to delegate separate busses for I/O, data memory space and program memory space. Multiple high-speed busses on a processor create other challenges, however. These challenges come in the form of excessive pin count and power requirements. Careful tradeoffs must be made to achieve the desired I/O performance, yet, at the same time, keeping the processor that meets the form factor and power capabilities for a range net-

work system designs.

Pipelined Pattern Matching

Consider the pattern matching functions required in the classification and forwarding blocks of a network processor. A variety of search algorithms exist which perform bit wise or bit field wise searches on a segment of a network address. Performing this sequentially has the advantage of a centralized lookup table, but the disadvantage of a significant number of processor cycles to achieve a result before the next search can commence. A search algorithm which lends it self to a pipelined approach would be desirable in the sense that each pipeline clock cycle produces a result and multiple lookups can be in process at any given time. To implement this requires an algorithm that, first of all, makes efficient use of lookup memory, and secondly lends itself to segmentation of lookup memory into discrete lookup blocks that can be sequenced in a pipelined fashion. The figure below shows a three-stage lookup which is implemented in a pipeline.

Given that a pattern search algorithm can be developed to efficiently be deployed in such a pipelined manner, several performance factors are greatly improved. First of all, memory bandwidth is distributed across lookup tables in each pipe stage, reducing memory bus speeds. Secondly, the clock speed of the pipeline is reduced significantly from an equivalent sequential (RISC) search engine. This increases the possibilities for scaling performance going forward.

Buffer and Queue Management

One of the more underestimated challenges in the network-processing arena is that of managing buffers and queues in a system. General purpose, programmable solutions have been non-existent due to the vast complexities that arise due to the range of network protocol and system architecture variants.

As more general purpose, programmable solutions appear on the market, a building block approach is more likely to be adopted by system designers due to several facts.

Key operations that must be performed in this area include Pointer Management, data byte counters, timers (for aging data) and queue servers capable of re-ordering data from ingress to egress.

Specialized hardware can be applied to these operations in order to improve performance. Large, high-speed register sets can be deployed to enable fast context switching and the storage and buffer pointer and counter/timer data.

The types of functions at the ingress and egress of a network device differ significantly enough that separate specialized processing elements are often required for ingress and egress queue management.

Data Modification

Network traffic requires a vast range of modifications within a network element. Minor bit field modifications and subsequent CRC/Checksum recalculations can be done fairly easily at line rate with specialized hardware. Packet data fragmentation and encapsulation require more processor intensive operations.

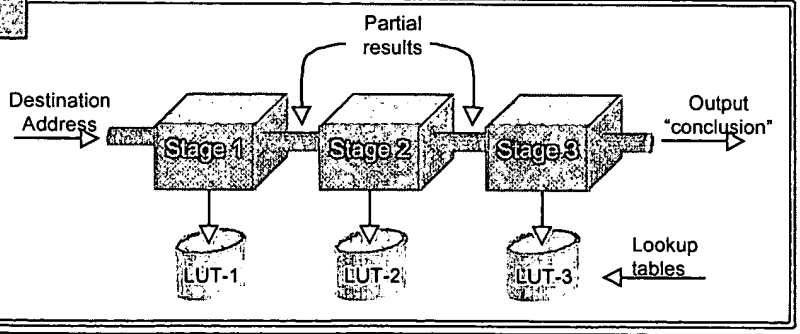
To develop a programmable, yet scalable architecture here requires defining the range of line rate operations to be performed. Specialized hardware processing blocks are invaluable in this area to offload any parallel sequential or pipelined processing architectures that can be deployed to handle the cases where a high degree of exception processing is required.

Statistics and Networks Data Gathering

Gathering network statistics, is a mundane task for a network processor, but requires forethought when scalability and programmability are desired. Gathered statistical data can be stored in memory or registers. The choice must be made weighing performance, pin count and line rate performance. A second factor must be taken into account as well. As large quantities of statistical and monitoring data

Pipelined Pattern Matching

Appropriately structured pattern matching algorithms can be pipelined to greatly increase throughput. In the diagram at the left, an IP destination address is processed in a 3-stage pipeline that generates intermediate results.



SWITCH FABRIC INTERFACE

is stored, the means to access it becomes an increasing challenge.

The control and access data path is a critical factor in developing a network processor that permits an external management device to access, configure and monitor network traffic statistical data.

Generally a network statistic processing engine can be deployed in parallel with other processing.

Putting It All Together

There's a vast range of options to be decided upon when choosing or developing a Network Processor topology. The diagram below attempts to show as generically as possible what the various operations are and where they exist in a system.

There are countless variations to the diagram below. Specifically in how the assorted function blocks are combined into programmable building blocks. Depending on the data rates involved, and how functions are partitioned, issues such as buffer size and memory and bus bandwidths must be resolved for each solution. Other challenges arise, as large quantities of data streams are processed, such as how ordering of incoming data is maintained on the egress.

In addition, system concerns arise when interfacing a programmable network processor that goes beyond just the data path bandwidth. Specifically, each system design has varying degrees of control data that is passed between the various functions within the system. For instance, a backplane or switch fabric interface may need communicate information to an ingress buffer manager with regards to scheduling data through it. An egress queue processor might need to advertise congestion on a port to the system to prevent queue overflow.

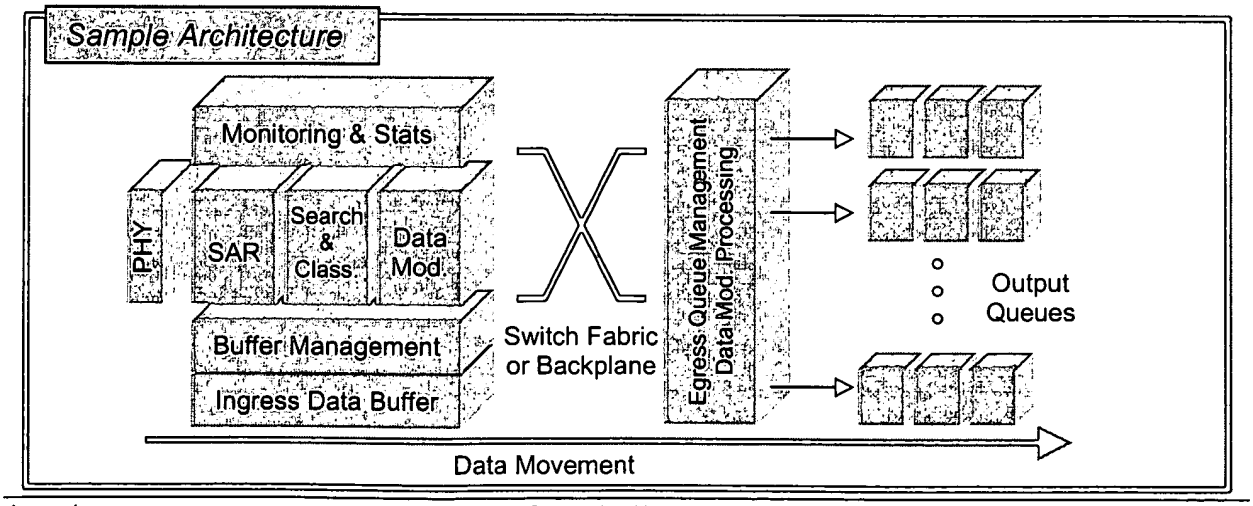
A flexible network processor must also take this control path into account and provide the flexibility in that path to enable the processor to be designed into a variety of system platforms.

Summary

To achieve both the flexibility and programmability of a general purpose processor, and maintain the performance required at gigabit speeds, one must look beyond Sequential, RISC processor architectures. New programmable engines are possible. These engines must combine pipelining, parallel processing and new algorithms deployed in hardware in innovative ways. A successful Network Processing architecture must take into account not only the performance requirements of today, but must also provide processing headroom to meet the scaling needs into the future.

Each proposed architecture for a Next Generation Network Processor must be scrutinized for it's scalability beyond it's current capabilities. Clock speeds, architecture complexity, gate count, and programming flexibility each must have sufficient headroom for growth if system designs are to be modified to accommodate these processors. In addition, these processors must have interface flexibility in order to be useful in more than just a small subset of system designs.

For more information contact Agere, at www.agere.com.



to create complex, custom-tailored applications to fit a variety of applications. However, due to the complex instructions that must be processed, CISC-based processors are difficult to scale to meet the increased performance demands that network applications require.

The alternative approach to CISC is called "Reduced Instruction Set Computing" or RISC. RISC processors are designed with support for a much simpler and smaller set of instructions that allows them to perform processing tasks at speeds much greater than their CISC counterparts, while still preserving much of the support for flexible programming that applications require.

Compared to ASICs, RISC processors are easily programmable. A RISC-based network processor follows the traditional software-based model and therefore retains the advantages (high level of programmability and flexibility) while also retaining the disadvantages (slower performance, since decisions are made in software) of that model.

The performance of RISC processors has been improved over the years by taking advantage of Moore's law (doubling of transistor density every 18 months). The decreased transisitor geometries allow both higher transisitor counts, and increased clock rates.

Most modern software-based routers use a RISC processing core (although in some configurations tasks are off-loaded to ASIC-based line cards). This RISC architecture makes these routers capable of supporting a large variety of protocols and features; however they do not scale well as additional features are added or additional bandwidth requirements are imposed. For example, activating features such as Access-Control Lists (ACL) or IP accounting on a RISC-based router under heavy load can cause the product to quickly reach its maximum processor utilization level, hurting traffic throughput. For more details on the architectural limitations of RISC Network Processors, please see the Agere white paper entitled "Building Next Generation Network Processors."

Parallel Processing

Most current CPUs are based upon the Von Neumann architecture, in which processors process information serially, one instruction at a time. Research continues into parallel processing in which processors can perform multiple instructions simultaneously.

For more details on the architectural limitations of parallel network processors, please see the Agere white paper entitled "Building Next Generation Network Processors".

Hybrid Approaches

For hardware vendors, the "Holy Grail" is the flexibility of RISC with the performance of ASICs. One approach that hardware vendors have tried in order to accomplish this is by using ASIC and RISC-based processors within the same product. In these types of devices, a central RISC based processors typically acts as the core processor and specific tasks are moved out to ASIC-based line cards.

An example of this is the Cisco Express Forwarding model in which the central processor calculates routes and downloads a complete copy of this routing information into line interface cards. The interface cards then use ASICs to switch traffic between themselves at very high rates of speed.

However this approach runs into the limitations previously described for ASIC-based architectures, namely that ASIC-based line cards can't be easily upgraded nor can they easily support new features.

This continues to leave hardware manufacturers in a quandary. The basic requirement continues: "how to combine the flexibility of RISC with the performance of ASICs" within a single platform. Fortunately, Agere has a solution.

"Wicked Smart"

Agere's Functional Processor

Agere's functional processor truly combines the flexibility of RISC with the performance of ASIC. The unique "Payload Plus" architecture uses a patented technology called "Pattern Matching Optimization" to achieve greater than 5x performance improvement over even advanced RISC processors. This performance reaches the level of fixed function ASICs while allowing the flexibility and programmability of RISC. The Payload Plus architecture is able realize these performance gains by using less overhead, fewer clock cycles, and more data processing per clock cycle than augmented RISC.

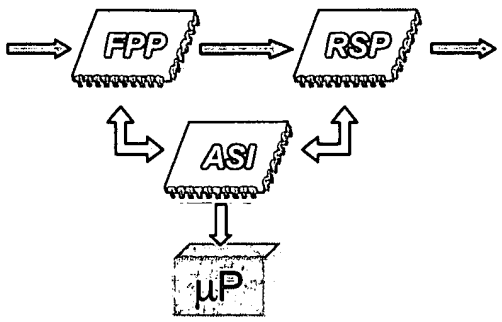
Compared to competitive products based on advanced RISC processors, only Agere's Payload Plus architecture is capable of supporting bandwidth speeds of OC-192 and beyond while still maintaining the flexibility and programmability of a traditional RISC processor.

The Agere functional processor is segmented into two components, the "Fast Pattern Processor" and the "Routing Switch Processor." The Fast Pattern Processor" (FPP) takes data from the PHY chip and performs protocol recognition and classification as well as reassembly. The FPP can classify traffic based on information contained at Layers 2 through 7. Based on parallel processing algorithms, the FPP can provide ATM reassembly and can support table lookups with millions of entries and variable entry lengths.

From the FPP, traffic moves to the Routing Switch Processor (RSP) which handles queuing, packet modification,

traffic shaping, application of quality of service tagging, and segmentation.

The FPP and RSP interface with the "Agere System Interface" (ASI). The ASI is the management component of the processor and provides support for RMON and tracks state information. The ASI also controls the movement of data between the FPP and RSP to ensure that it moves at wire speed rates.



Application of Agere's Functional Processor

In a typical system architecture, functions are partitioned into two types of processors. A general-purpose RISC processor can be used to develop and maintain the Routing Information Base (RIB) – the RISC processor calculates routes and handles OSPF, MPLS LDP, or RSVP packets and signaling. These are control functions where high processing rates are not required.

In contrast, the Agere Functional Processor is best applied to the data "fast path," and thus helps network equipment providers to offer products capable of true wire-speed performance. In a typical system, the RISC processor will forward RIB information to a Forwarding Information Base (FIB) located in each of the line cards. Using Agere high-performance FPP/RSP technology, these line cards are can enforce policies and forward packets at very high speeds. However, unlike ASICs, Agere functional processors are programmable and can flexibly accommodate product changes.

Agere's Functional Programming Language

In order to allow hardware manufacturers to quickly develop and build solutions based on Agere's functional processor, Agere provides a rich development environment called the "Functional Programming Language" (FPL). This language is a true fourth-generation programming language that is designed to be highly intuitive. Instructions are coded like a protocol definition language, with interspersed action statements and the ability to embed routing table information directly into the protocol code.

Agere's FPL provides developers with an enormous advantage over complex to develop RISC code and allows developers to shorten time-to-market cycles thus reducing life-cycle development expenses. Through its functional approach, FPL provides a high level of reuse of code in multiple applications.

Agere's programming environment also delivers a full suite of testing and simulation tools that allow developers to fully test applications before deployment. Agere's "Integrated Development Environment" (IDE) provides developers with a complete software developer's kit including traffic generation tools for real-world simulation and testing.

For more information on the advantages of FPL, see the Agere white paper entitled "The Case for a Classification Language."

Summary

The demands places on network processors are expected to rapidly scale over the next several years due to demand for bandwidth as well as demands by carriers to provide new services. Current network processor technology is ill equipped to adequately meet these demands. Current ASIC processors aren't flexible enough to support a rapidly changing marketplace, while RISC and augmented RISC processors will not be able to scale to support the throughput rates required for complex features at speeds of GbE, OC-48, and above, and require complex programming methods. Only Agere's "Wicked Smart" approach provides network processors that combines the performance of ASICs with the flexibility and feature support of RISC, exploited by a comprehensive and simple development environment. This unique combination will allows network equipment manufacturers to quickly bring new products to market that can scale to support ever increasing bandwidth demands, while supporting complex feature sets for Quality of Service and SLA management.

By exploiting the advantages of Agere's approach, manufacturers will be able to shorten time-to-market cycles, create new products that scale to meet future performance demands, and quickly add new features as market conditions warrant. Using Agere's network processors, equipment manufacturers will realize a tremendous competitive advantage over competitors using alternate approaches.

For more information, contact Agere at www.agere.com.

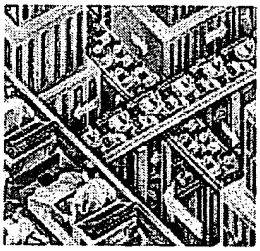
Feature	Comment/Description	Benefit
Rich ATM Traffic Management Mechanisms		
Multiple Priorities for Queuing and Scheduling	Supports all ATM Forum traffic classes and AAL types	Supports all traffic types and allows for QoS on demand
Multiple Queue Scheduling Disciplines for Frames and Cells	Strict Priority	Custom, guaranteed ATM traffic classes
	Specified Rate	Guarantees peak rate for CBR (rate limiting), minimum cell rate for ABR or UBR, and sustainable cell rate for VBR
	Weighted Round Robin (WRR)	Fair, differentiated delay probability for UBR or NRT-VBR traffic classes, and for best effort with multiple IP precedence levels or RSVP service classes in Tag Switching networks
Intelligent Early Packet Discard	Supports early packet drop when user-defined buffer threshold(s) are exceeded	Maximizes goodput for bursty traffic; multiple thresholds used for weighted early packet discard offer differentiated loss probability for UBR and for best effort with multiple IP precedence levels or RSVP IP service classes in MPLS/Tag Switching networks
Intelligent Tail (Partial) Packet Discard	Drops remaining cells of an AAL5 frame when any cells are dropped after failing traffic policing, or any other event that causes cell loss	Improves goodput by not needlessly forwarding cells of a partial frame and increases buffer utilization, thus avoiding any further loss
Cell Tagging and Selective Cell Drop	Sets cell loss priority bit in cells that exceed buffer thresholds or fail traffic policing	Optimizes network utilization while minimizing unnecessary cell loss
	Preferentially drops cells with CLP bit set	Prioritizes dropping of non-conformant cells
Traffic Policing (usage parameter control) per ITU-T I.371 and ATM Forum UNI Specifications	Uses dual leaky bucket algorithm	Allows enforcement of traffic contracts for all ATM Forum-defined traffic types
Traffic Pacing/Shaping	Allows cells to be paced out specific connections and shaped VP tunnels at rates equal tor below the line rate	Allows for rate-limited traffic on public network connections; 128 total VP tunnels can be shaped per switch, each VP with up to 128 VCs each
Resource and Policy Connection Admission Control	Used in admitting and routing ATM connections	Ensures that guarantees made to existing connections or policies set by the administrator are honored when setting up new connections
ABR Congestion Control	Supports EFCI and Relative Rate (RR) Marking on Catalyst 8510 MSR	Minimizes cell loss and maximizes goodput
Traffic Statistics	Wide variety of per-connection, per-port, and per-switch statistics	Allows full monitoring of switch and connection performance
LAN Emulation Services	LANE Configuration Server (LECS), LANE Server (LES), and Broadcast and Unknown Server (BUS) for Ethernet and Token Ring emulated LANs (supporting LUNI v1.0 or v2.0 clients); the LECS supports MPOA networks as well	Deployment option for the backup LANE servers in Cisco's unique Simple Server Redundancy Protocol (SSRP) for LANE
Complete Network Redundancy	Cisco's standards-based SSRP and Hot Standby Router Protocol (HSRP) for LANE and MPOA networks	Provides redundancy for the LANE servers (via SSRP) and the default inter-VLAN router and MPOA server (via HSRP)
RFC 1577 Classical IP over ATM	Supports ATM ARP server	A deployment option
ATM Forum CES	Integrates voice and video with data traffic onto a single ATM trunk Structured (n x 64) and Unstructured (Clear Channel) Services	Used to interconnect PBXs, TDMs, and typical site-to-site videoconferencing equipment, potentially avoiding local leased-line charges and saving monthly costs
	On-hook suppression	Frees CBR bandwidth for other traffic

PACKET
SECOND QUARTER 2000

▼ SECTIONS

Implementing QoS for Packet Telephony

By Bruce Davie



One of the most important challenges in supporting telephony over IP and other packet networks is to meet the stringent quality-of-service (QoS) requirements of interactive voice communication. User perceptions of voice quality depend heavily on the timely delivery of voice-over-IP (VoIP) packets and, to a lesser extent, on low packet-loss rates.

There are two major alternatives for enterprises and public network operators to implement QoS for voice: Differentiated Services (Diff-Serv) and Integrated Services (Int-Serv). Diff-Serv is a *coarse-grained* approach, providing QoS to aggregated traffic, while Int-Serv is a *fine-grained* approach that provides QoS to individual applications or flows. These two approaches have been standardized in the Internet Engineering Task Force (IETF), and both are supported in Cisco IOS® software.

- Diff-Serv (RFC 2474)
- Diff-Serv (RFC 2475)
- The scalable application of RSVP to high-speed backbone networks: (RSVP aggregation) and (RSVP operation over Diff-Serv networks)
- See also *Inside the Internet's Resource ReSerVation Protocol: Foundations for Quality of Service* by David Durham and Raj Yavatkar

So far, we have seen two components of the Integrated Services architecture: signaling (as accomplished by RSVP) and admission control. Other components of the architecture include *classification, policing,* and *scheduling.*

Classification is the process of identifying packets that belong to flows (calls) for which reservations have been made, so that they can receive the appropriate QoS. Whereas this classification in a Diff-Serv network may be done using just a few bits in the IP header, Int-Serv classification may examine up to five fields in the packet: the source address, destination address, protocol number, source port, and destination port. Based on this information, packets can be given the appropriate policing and scheduling treatment.

Policing is the function applied to packets in a reserved flow to ensure that it conforms to the traffic specification (TSpec) that was used to make the reservation. If a flow does not conform -- because the source is sending more bytes per second than it advertised or for other reasons -- then it is likely to interfere with the service provided to other flows, and corrective action is needed. Options include dropping excess packets or giving them lower priority in the scheduler. Admission control takes place at call setup time to see if a reservation can be allowed, while policing is applied to each packet at forwarding time to ensure that the flow is behaving correctly.

The final ingredient is *scheduling*. Many scheduling algorithms can be used to support Int-Serv. Weighted Fair Queuing (WFQ) is one popular algorithm that can provide a guaranteed service rate to each reserved flow. Another possibility is to place all the packets from reserved flows in a single high-priority queue to ensure that they are transmitted ahead of best-effort packets. Both algorithms

Next Up Previous Contents

Next: RSVP protocol overview **Up:** Types of service in **Previous:** Controlled load service

Implementation of traffic control

Every network element on a path has to support the distinction between different services. Otherwise, it is not guaranteed that the demanded end-to-end quality of service can be supported.

It is left to the router vendors to implement the specified services in network elements. Implementations may differ in the underlying mechanisms, processing speed and utilization of the bandwidth.

A possible implementation framework is given by the Integrated Services working group. The scheduling algorithm *weighted fair queuing* (WFQ) is an important building block. WFQ is used to isolate flows from each other. Each WFQ flow has a separate queue and packets are scheduled so that each flow receives a constant fraction of the link bandwidth during congestion. In contrast to static time-slicing mechanisms, no bandwidth is wasted while there is demand. If a WFQ flow does not use its assigned bandwidth, other flows can use it. Compared to weighted round robin, WFQ is computationally more expensive because it dynamically determines the next queue to be served according to the bandwidth that each queue received previously. Weighted round robin is only fair in terms of packets sent by each flow, not in terms of bandwidth used by each flow. For that reason, weighted round robin cannot be used to isolate flows, and the more complex WFQ algorithm is used.

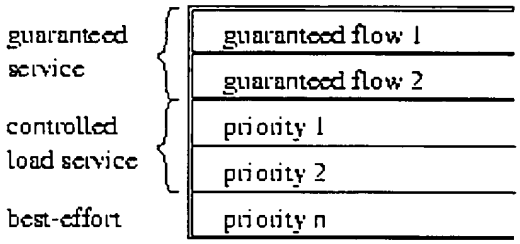


Figure 2.3: Hierarchical traffic control

At the top level, each guaranteed service flow gets its own WFQ queue. Thus, guaranteed flows are strictly separated from each other and from the rest of the traffic. All other traffic is assigned to a pseudo WFQ flow. Within this flow, controlled load traffic and best-effort traffic are separated by priority. The network only admits a certain amount of controlled load traffic to ensure that best-effort service is not completely preempted. Within the controlled load class, subclasses with different delay bounds are provided, separated by priority. To clear bursts, a higher-priority class can temporarily borrow bandwidth from a lower-priority class. Thus, priorities allow sharing of bandwidth in one direction and isolation in the other direction. Within each subclass, the overall delay is minimized by simple FIFO scheduling. Flows within a subclass should all have similar traffic characteristics, so when there is a burst in one flow, the other flows can share the delay without being too much delayed themselves. Figure 2.3 illustrates an example of this hierarchical approach.

Next Up Previous Contents

Next: RSVP protocol overview **Up:** Types of service in **Previous:** Controlled load service

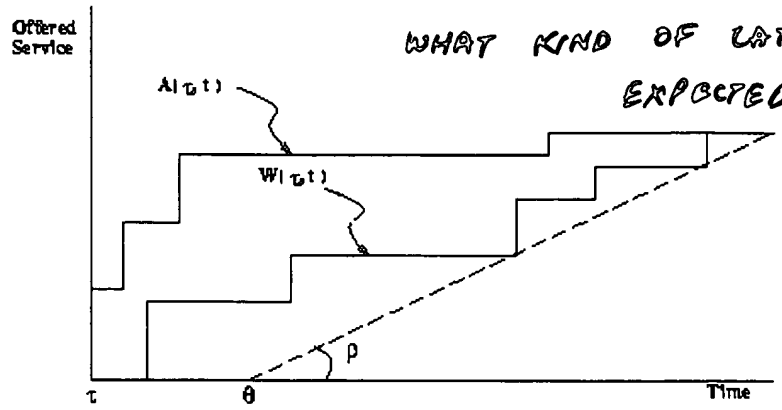


Figure 3.2: An example of the behavior of an $\mathcal{LR}$ scheduler.

beginning of its busy period is lower-bounded. Thus, for a scheduling algorithm to belong to this class, it is only required that the *average* rate of service offered by the scheduler to a busy session, over every interval starting at time Θ_i^S from the beginning of the busy period, is at least equal to its reserved rate. This is much less restrictive than GPS multiplexing, where the *instantaneous* bandwidth offered to a session is bounded. That is, the lower bound on the service rate of GPS multiplexing holds for any interval $(\tau, t]$ that a session is backlogged, whereas in $\mathcal{LR}$-servers the restriction holds only for intervals starting at the beginning of the busy period. Therefore, GPS multiplexing is only one member of the $\mathcal{LR}$ class.

The latency parameter depends on the scheduling algorithm used as well as the allocated rate and traffic parameters of the session being analyzed. For a particular scheduling algorithm, several parameters such as its transmission rate on the outgoing link, number of sessions sharing the link, and their allocated rates, may influence the latency. However, we will now show that the maximum end-to-end delay experienced by a packet in a network of schedulers can be calculated from only the latencies of the individual schedulers on the path of the session, and the traffic parameters of the session that generated the packet. Since the maximum delay in a scheduler increases directly in proportion to its latency, the model brings out the significance of using low-latency schedulers to achieve low end-to-end delays.

Proposition 3.1: *The end-to-end delay and increase in burstiness of a session in a network of $\mathcal{LR}$ servers is proportional to the latency Θ_i^S of the servers. We can minimize both of these parameters by designing servers with minimum latency.*

Note that the latency of a server depends, in general, on its internal parameters and the bandwidth allocation of the session under consideration. In addition, the latency may also vary with the number of active sessions and their allocations. Such a dependence of the latency of one session on other sessions indicates the poor isolation properties of the scheduler. Likewise, in some schedulers the latency may depend heavily on its internal parameters, and less on the bandwidth allocation of the session under observation. Such schedulers do not allow us to control the latency of a session by controlling its bandwidth allocation. On the other hand, the latency of a PGPS server depends heavily on the allocated bandwidth of the session under consideration. This flexibility is greatly desirable.

3.1.3 Delay Bound for Dual-Leaky Bucket Shaped Sources

Since the definition of an $\mathcal{LR}$ server is not based on any assumptions on the input traffic, it is easy to derive delay bounds for traffic distributions other than the (σ, ρ) model. For example, when the peak rate of the source is known, a modified upper bound on the delay of an $\mathcal{LR}$ server can be obtained. Let us denote with g_i the service rate allocated to connection i, and let ρ_i and P_i respectively denote the average and peak rate at the source of connection i. The arrivals at the input of the server during the interval $(\tau, t]$ now satisfy the inequality

$$A_i(\tau, t) \leq \min\left(\sigma_i + \rho_i(t-\tau), P_i(t-\tau)\right). \tag{3.42}$$

We can prove the following lemma:

Lemma 3.6: *The maximum delay D_i of a session i in an $\mathcal{LR}$ server, where the peak rate of the source is known, is bounded as*

$$\left(\frac{P_i - g_i}{\quad}\right)\left(\frac{\sigma_i}{\quad}\right)$$

In Lemma 3.7 we proved that, if we use the backlogged period to bound the service offered by a server S, then S is an $\mathcal{LR}$ server and its latency can not be larger than that found for the backlogged period. However, we must emphasize the fact that the opposite is not true. Consider the example of Figure 3.4. Let us assume an $\mathcal{LR}$-server with rate ρ and latency Θ. Referring to Figure 3.4, time-intervals $(0, t_1]$ and $(t_2, t_3]$ form two busy periods. However, the server remains backlogged during the whole interval $(t_1, t3]$. If the backlogged period was used to bound the service offered by the server, a latency $\Theta_b > \Theta$ would result. By extending the above example over multiple busy periods, it is easy to verify that Θ_b can not be bounded. This shows that if backlogged period was used instead of busy period in the definition of the $\mathcal{LR}$ server model, the end-to-end delays of the server would not be bounded.

By using the above lemma as our tool, we can analyze several work conserving servers and prove that they belong in the class $\mathcal{LR}$. In the following proofs, L_i denotes the maximum packet-size for session i and L_{max} the maximum packet-size among all sessions.

3.2.1 Weighted-Fair-Queueing

We first start by showing that a GPS scheduler is an $\mathcal{LR}$ server.

Lemma 3.8: *A GPS scheduler belongs to the class $\mathcal{LR}$ and its latency is zero.*

Proof: From the definition of the GPS multiplexer, during any interval of time that connection i is continuously backlogged,

$$W_i^{GPS}(\tau, t) \geq \rho_i(t - \tau).$$

From Lemma 3.7 it easy to conclude that a GPS scheduler is an $\mathcal{LR}$-server with zero latency.

□

Weighted-Fair-Queueing or Packet-by-packet GPS (PGPS) scheduling algorithm is the packet-by-packet equivalent of GPS multiplexing. In [70] it was proven that, if t^F, t^P are

Server	Latency
GPS	0
PGPS	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$
SCFQ	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}(V-1)$
VirtualClock	$\frac{L_i}{\rho_i} + \frac{L_{max}}{r}$
Deficit Round Robin	$\frac{3F-2\phi_i}{r}$
Weighted Round Robin	$\frac{F-\phi_i+L_c}{r}$

Table 3.1: Some $\mathcal{LR}$ servers and their latencies. L_i is the maximum packet size of session i and L_{max} the maximum packet size among all the sessions. In weighted round-robin and deficit round-robin, F is the frame size and ϕ_i is the amount of traffic in the frame allocated to session i. L_c is the size of the fixed packet (cell) in weighted round-robin.

This is essential the latency of a cell that arrives in the beginning of a round and is serviced at the end of the round. It can be easily verified that this bound for the latency is tight.

□

A summary of our results is presented in Table 3.1. It is easy to see that PGPS and VirtualClock have the lowest latency among all the servers. In addition, their latency does not depend on the number of connections sharing the same outgoing link. As we will show in Section 3.4, however, that VirtualClock is not a fair algorithm.

In self-clocked fair queueing, the latency is a linear function of the maximum number of connections sharing the outgoing link. In Deficit Round Robin, the latency depends on the frame size F. By the definition of the algorithm, the frame size, in turn, is determined by the granularity of the bandwidth allocation and the maximum packet size of its session. That is,

$$\sum^{V} L_i \leq F,$$

THIS PAGE BLANK (USPTO)

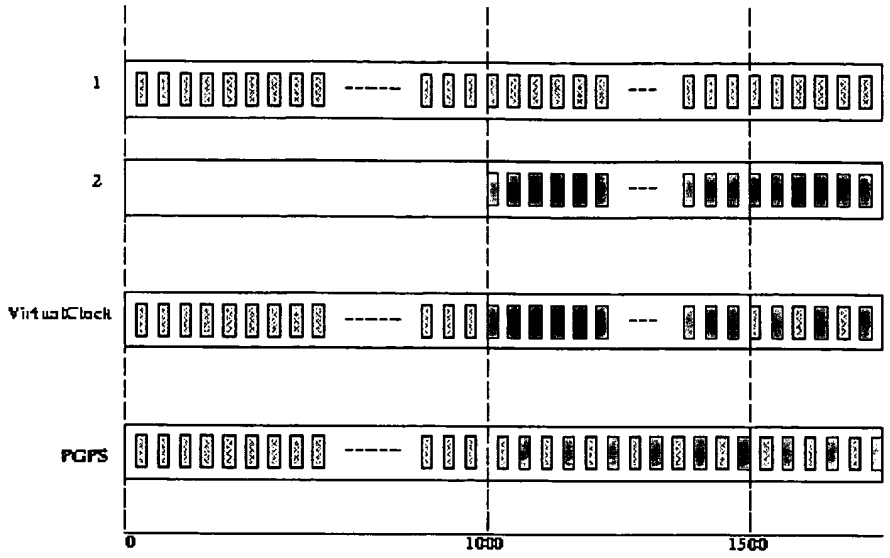


Figure 3.6: Unfair behavior of the VirtualClock scheduling algorithm. The two connections are allocated equal portions of the outgoing link bandwidth.

algorithm used. We use as a measure of fairness the difference in normalized service offered to the two connections for any time interval $(t_1, t_2]$ after time τ.

A typical example of unfairness occurs in the VirtualClock algorithm, as illustrated in Figure 3.6. Assume that two connection share an outgoing link and are allocated equal shares of the link bandwidth. Assume each packet is of unit size and the rate of the server is also unity. Consider an interval of time 0–1000 during which only connection 1 is active, and sends 1000 packets. Connection 2 becomes active at time 1000, and both connections send packets after 1000. Assume that the scheduler is based on the VirtualClock algorithm. Since the server is work-conserving, it will service all 1000 packets from connection 1 by $t = 1000$. However, at time $t = 1000$, the next arriving packet of connection 1 will receive a timestamp of 2000, reflecting the average service received by the connection until 1000. Thus, connection 1 will be starved until time 1500, when the timestamps of the packets of the two connections become equal. If the maximum burstiness of the sources is not bounded, we can see that the interval over which a backlogged connection is denied service can grow to infinity. A PGPS server, in contrast, provides equal service to the two connection after $t = 1000$ regardless of the excess bandwidth received by the connection 1 earlier.

BUFFER SIZE

DO WE NEED IN DEEP PACKET

4. Rate-Proportional Servers: A Design Methodology for Fair Queueing Algorithms

BUFFER WITH WFQ?

As we presented in the previous chapter, the packet-by-packet approximation of GPS (WFQ) has the lowest latency among all the packet-by-packet servers; thus, from Theorem 3.1, WFQ has the lowest bounds on end-to-end delay and buffer requirements. However, WFQ also has the highest implementation complexity. VirtualClock has the same latency as WFQ, but is not a fair algorithm [70, 104]. Self-Clocked Fair Queueing and the round-robin schedulers provide bounded unfairness, but their latency is a function of the number of connections that share the output link. In a broadband network, the resulting end-to-end delay bounds may be prohibitively large.

A scheduling algorithm that combines the delay and burstiness behavior of Weighted Fair Queueing, simple timestamp computations, and bounded unfairness, has so far remained elusive. In this chapter we develop an analytical framework for the design of such algorithms, and systematically analyze its properties; later in Chapter 5 we will present scheduling algorithms based on this framework that have simple and efficient implementations.

Thus, in this chapter we present a general class of schedulers, that we call *Rate-Proportional Servers* (RPS). Schedulers in the RPS class offer the same end-to-end delay and burstiness bounds as WFQ. Since the class of rate-proportional servers is based on a general definition, multiple algorithms with the same properties but with different implementation complexities may be designed. Depending on their design, schedulers in the RPS class may have substantially different fairness properties. It is shown that both GPS, an algorithm with ideal fairness, and a fluid-model equivalent of VirtualClock, an unfair algorithm, are members of the RPS class.

The rest of this chapter is organized as follows: In Section 4.1, we present the concept of the potential function that is the basic mechanism for designing such algorithms. In

PRIORITY

HOW MANY PRIORITIES CAN THE DEVICE HANDLE

From the above definition of potentials, it is clear that a fair algorithm must attempt to increase the potentials of all backlogged connections at the same rate, the rate of increase of the system potential. Thus, the basic objective is to *equalize the potential* of each connection. Sorted-priority schedulers such as GPS, WFQ, SCFQ, and VirtualClock all attempt to achieve this objective. However, in our definition of potential, we did not specify how the potential of a connection is updated when it is idle, except that the potential is non-decreasing. Scheduling algorithms differ in the way they update the potentials of idle connections. Ideally, during every time interval that a connection i is not backlogged, its potential must increase by the normalized service that the connection could receive if it were backlogged. If the potential of an idle connection is increased by the normalized service it missed, it is easy to see that, when the connection becomes busy again, its potential will be identical to that of other backlogged connections in the system, allowing it to receive service immediately.

One way to update the potential of a connection when it becomes backlogged is to define a *system potential* that keeps track of the progress of the total work done by the scheduler. The system potential $P(t)$ is a non-decreasing function of time. When an idle session i becomes backlogged at time t, its potential $P_i(t)$ can be set to $P(t)$ to account for the service it missed. Schedulers use different functions to maintain the system potential, giving rise to widely different delay- and fairness-behaviors. In general, the system potential at time t can be defined as a non-decreasing function of the potentials of the individual connections before time t, and the real time t.

$$P(t) = \mathcal{F}(P_1(t-), P_2(t-), \ldots, P_V(t-), t). \tag{4.1}$$

For example, the GPS server initializes the potential of a newly backlogged connection to that of a connection currently backlogged in the system. That is,

$$P(t) = P_i(t), \quad \text{for any } i \in B(t);$$

and therefore,

$$\begin{aligned} P(t_2) - P(t_1) &= P_i(t_2) - P_i(t_1) \\ &= \frac{W_i(t_1, t_2)}{\rho_i} \\ &\geq t_2 - t_1. \end{aligned}$$

VirtualClock is a rate-proportional server as well. Consider a server where the system potential function is defined as

$$P(t) = t.$$

It is easy to verify that such a server satisfies all the properties of a rate-proportional server. Consider a packet-by-packet server that transmits packets in increasing order of their finishing potentials. Such a server is equivalent to the packet-by-packet VirtualClock server.

We now proceed to show that every rate-proportional server is a zero-latency server. This will establish that this class of servers provide the same upper-bounds on end-to-end delay as GPS. To prove this result, we first introduce the following definitions:

Definition 4.2: *A session-i* **active** *period is a maximal interval of time during a system busy period, over which the potential of the session is not less than the potential of the system. Any other period will be considered as an* **inactive** *period for session i.*

The concept of active period is useful in analyzing the behavior of a rate-proportional scheduler. When a connection is in an inactive period, it can not be backlogged and therefore can not be receiving any service. On the other hand, an active period need not be the same as a backlogged period for the connection. Since, in a rate-proportional server, the potential of a connection can be below the system potential only when the connection is idle, a transition from inactive to active state can occur only by the arrival of a packet of a connection that is currently idle, whose potential is below that of the system.

CONNECTIONS

It is clear that no arrivals of session-i packets occurred during the interval $(t^* - \Delta t, t^*)$; if there was an arrival during this interval, the connection would have entered an active period. Thus,

$$A_i(\tau, t^* - \Delta t) = A_i(\tau, t^*). \tag{4.23}$$

From equations (4.16), (4.21), (4.22), and (4.23) we can conclude that

$$A_i(\tau, t^*) < \rho_i(t^* - \tau). \tag{4.24}$$

This means that time t^* does not belong in the same busy period as $t^* - \Delta t$. □

Thus, the definition of rate-proportional servers provides us a tool to design scheduling algorithms with zero latency. Since both GPS and VirtualClock can be considered as rate-proportional servers, by Theorem 4.1, they have the same worst-case delay behavior.

4.2.1 Packet-by-Packet Rate-Proportional Servers

In the previous section we defined the rate proportional servers using a fluid-model, where packets from different connections can be served at the same time with different rates. However, in a real system only one connection can be serviced at each time and in addition packets can not be split in smaller units. A packet-by-packet rate proportional server can be defined in terms of the fluid-model as one that transmits packets in increasing order of their finishing potential. Let us assume that when a packet from connection i finishes service in the fluid server, the potential of connection i is TS_i. We can use this finishing potential to timestamp packets and schedule them in increasing order of their time-stamps. We call such a server a *packet-by-packet rate-proportional server* (PRPS).

In the following, we denote the maximum packet size of session i as L_i and the maximum packet size among all the sessions as L_{max}.

In order to analyze the performance of a packet-by-packet rate-proportional server we will bound the difference of service offered between the packet-by-packet server and the

Since Weighted Fair Queueing is a packet-by-packet rate proportional server with $\Delta P = 0$, we obtain the following result on the fairness of a WFQ scheduler by setting $\Delta P = 0$ in Eq. (4.54).

Corollary 4.3: *For a WFQ scheduler,*

$$\left| \frac{\hat{W}_j(t_1, t_2)}{\rho_j} - \frac{\hat{W}_i(t_1, t_2)}{\rho_i} \right| \leq \max(C_j + \frac{L_{max}}{\rho_i} + \frac{L_j}{\rho_j}, C_i + \frac{L_{max}}{\rho_j} + \frac{L_i}{\rho_i})$$

It can be shown that the above bound is tight. For example, consider the case where connections i, j are already backlogged in the system, at time τ. Then, connection j may have received an additional amount of service equal to C_i in the WFQ server and connection i may have received less service equal to L_{max} in the WFQ server compared to the GPS server. The finishing potential of the last packet serviced from connection i before a packet is serviced from connection j may be $F_i \leq P_j(\tau) + C_i + \frac{L_i}{\rho_i}$. The total normalized service that connection i may receive while connection j is waiting is bounded by $C_j + \frac{L_j}{\rho_j} + \frac{L_{max}}{\rho_i}$.

4.4 Summary

In this chapter we developed the framework of rate-proportional servers (RPS) for designing schedulers with low latency and bounded unfairness. Fundamental to the definition of rate-proportional servers is the system-potential function that maintains the global state of the system by tracking the service offered by the system to all connections sharing the outgoing link. Similarly, the state of each connection is represented by a connection potential function. In a packet-by-packet server, the system potential and connection potentials provide the basis for computing a timestamp for each arriving packet. Packets are then transmitted in increasing order of their timestamps. We defined the necessary properties the system potential function must satisfy for the server to provide a delay bound, burstiness and buffer requirements identical to that of Weighted Fair Queueing.

Besides providing valuable insight into the behavior of scheduling algorithms, the RPS

class, with application in both general packet networks and in ATM networks. Note that the fundamental difficulty in designing a practical rate-proportional server is the need to maintain the system potential function. Tracking the global state of the system precisely requires simulating the corresponding fluid-model RPS in parallel with the packet-by-packet system. The algorithms in the next chapter, however, avoid this need by maintaining the system potential only as an approximation of the actual global state in the fluid model, and re-calibrating the system potential periodically to correct any discrepancies. In the Frame-based Fair Queueing algorithm this re-calibration is done at frame boundaries, while in Starting Potential-based Fair Queueing (SPFQ) the re-calibration occurs at packet boundaries. This gives rise to two algorithms with the same delay bound, but with slightly different fairness properties. Both algorithms, however, provide bounded unfairness and $O(1)$ timestamp computations.